Computer persönlich

Band 1:

Ein Leitfaden für den Erstanwender

Das Commodore 64 Buch

Hans Lorenz Schneider • Werner Eberl

Mit Assembler und Basic-Erweiterungen für hochauflösende Grafik

Das Commodore 64-Buch

Hans Lorenz Schneider
Werner Eberl

Das Commodore 64-Buch

Band 1: Ein Leitfaden für den Erstanwender

Verlag Markt & Technik

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Schneider, Hans Lorenz:
[Das Commodore-vierundsechzig-Buch]
Das Commodore-64-Buch / Hans Lorenz Schneider ;
Werner Eberl. — Haar bei München : Verlag Markt u. Technik
(Computer persönlich)
NE: Eberl, Werner:
Bd. 1. Ein Leitfaden für den Erstanwender. — 1984.
ISBN 3-922120-61-X

»Commodore 64« ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt, die ebenso wie der Name »Commodore« Schutzrechte genießt. Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis der Schutzrechtsinhaberin.

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

ISBN 3-922120-61-X

© 1984 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder-Druck, 8906 Gersthofen
Printed in Germany

Vorwort

Einige Bücher die den Commodore 64 zum Thema haben, sind schon auf dem Markt erhältlich. Die meisten Bücher sind jedoch für die Belange von bestimmten Anwendergruppen geschrieben worden: So gibt es Bücher die sich mit den spezifischen Internas des Commodore 64 beschäftigen, die jedoch nur dem Computer-Enthusiasten, der schon andere Maschinen kennt, bei seiner Arbeit unterstützen. Andere Bücher beschäftigen sich z.B. nur mit Spezialgebieten des Commodore 64, wie der hochauflösenden Grafik.

Das vorliegende Buch soll für den Einsteiger mit dem Commodore 64 eine Unterstützung sein. Das heißt nicht, daß dieses Buch einen kompletten Basic-Kurs beinhaltet, sondern daß von Beginn an alle Möglichkeiten des Commodore 64 beschrieben werden. Andere Bücher sind durchaus zur weiteren Lektüre empfohlen, wie z.B. die Buchreihe 'Basic ohne Probleme'.

Im ersten Kapitel werden zur Einführung einige Programme beschrieben, die in dieser Form auch auf anderen Rechnern (besonders anderen Commodore-Rechnern) laufen. Erst nach dieser Grundlage wird auf die speziellen Eigenschaften des Commodore 64 eingegangen.

Die meisten Programme sind so aufgebaut, daß sie vom Leser noch nach seinen Wünschen ergänzt werden können. Für Übungszwecke werden immer wieder Tips für Änderungen oder Erweiterungen gegeben. In einigen Programmen (wie z.B. dem Programm für geometrische Berechnungen) wurden für diese Übungszwecke sogar der Platz freigehalten. Wie man daran leicht erkennen kann, ist nicht der Sinn dieses Buches, vorgefertigte Programme zu liefern, die ja auch auf Diskette lieferbar sind, sondern den Leser beim aktiven Arbeiten mit seinem Rechner zu unterstützen.

Das Autorenteam würde sich freuen, wenn Sie ihm Ihre Meinung zu dem Buch mitteilen würden. Dies kann in Form von Tips und Anregungen sein, aber auch konstruktive Kritik ist erwünscht.

Bis auf Kapitel 5 - über Assembler selbst kann man ganze Bücher schreiben - wurde hohen Wert auf ausreichende Dokumentation gelegt. In diesem Rahmen kann man nicht alle

Eigenschaften des Commodore 64 beschreiben. Deshalb wird u.a. die Beschreibung des Sound-Generators und der mehrfarbigen Grafiken in Band 3 vorgenommen. Band 2 wird sich ausschließlich mit Spielen beschäftigen. Aber nicht nur Spiele 'zum Abtippen' sondern zum eigenen Verfeinern. Wie in diesem Buch, werden auch in den weiteren lauffähige Programme zusammengestellt und mit Tips für Änderungen und Ergänzungen versehen.

Nicht zuletzt wollen wir uns auch bei Frau Marga Strauss bedanken, unserem sympatischen 'Textautomaten'. Ohne Sie wäre das Buch sicher nicht so schnell zustande gekommen.

München im November 1983

Hans Lorenz Schneider

Werner Eberl

Einleitung

Wie im Vorwort bereits erwähnt, befaßt sich das vorliegende Buch mit Übungsprogrammen für den Computerneuling, aber auch die Computer-Freaks kommen nicht zu kurz.

Im ersten Kapitel werden einige mathematische Programme besprochen, die zur Eingewöhnung an den Rechner und seine Handhabung dienen sollen. Von einer relativ einfachen mathematischen Operation (Matrizenmultiplikation) über noch einfachere geometrische Berechnungen - wo mehr Wert auf hierarchischen Menüaufbau gelegt wird - bis hin zu einem Grundgerüst für Statistikprogramme reicht hier die Palette.

Das nächste Kapitel ist den Grafikzeichen gewidmet, nicht zu verwechseln mit der hochauflösenden Grafik. Hier werden zwei Programme vorgestellt, die mit Spielkarten zu tun haben und einige Programmen zum Generieren von Balkendiagrammen. Die bis dahin vorgestellten Programme laufen ebenso auf anderen Commodore-Rechnern, wenn auch nicht mit unterschiedlichen Farben.

Das dritte Kapitel widmet sich dann einer Besonderheit des Commodore 64: den Sprites. Da das Errechnen der benötigten Daten für die Sprites sehr mühsam ist, werden zunächst zwei Programme vorgestellt, mit denen man sich diese Sache etwas erleichtern kann. Im Folgenden wird darauf eingegangen, wie diese Sprites am Rechner verwaltet werden, wie sie bewegt werden, und wie man die Berührung von Sprites abprüfen kann. Den Abschluß bildet eine Zusammenfassung, in der auch Tips für die Erweiterung des in Kapitel 3 durchgehend bahandelten Spieles gegeben werden.

Kapitel 4 beschäftigt sich mit der hochauflösenden Grafik, wobei zunächst auf die rechnerischen Internas des Commodore 64 eingegangen wird, die unbedingte Voraussetzung zum Verständnis der darauf folgenden Programme sind. Dann werden einige Unterprogramme vorgestellt, mit denen häufige geometrische Formen dargestellt werden können, sowie je ein Programm für Tortengrafik und Balkendiagramme. Es ist nur natürlich, daß mit Balkendiagrammen in hochauflösender Grafik eine feinere Darstellung möglich ist, wenn auch nicht eine bessere Auflösung. So werden doch z.B. die Balken auch als Quader dargestellt, um ein räumliches Bild der Grafik zu bieten.

Das Kapitel 5 ist - wie auch Kapitel 6 - für die Leser gedacht, die tiefer in den Commodore 64 einsteigen wollen, und sich deshalb auch nicht scheuen, Programme im Maschinencode zu schreiben. Ein Hilfsmittel für die Programmierung im Maschinencode ist der vorgestellte Assembler, der selbst wieder ein Basic-Programm ist.

Kapitel 6 beschäftigt sich mit Basic-Erweiterungen, die zwangsläufig in Assembler programmiert sind. Wer sich bei den Grafikzeichnungen in Kapitel 4 über den hohen Rechenzeitaufwand geärgert hat, der wird sicherlich gern auf die in Kapitel 6.2 beschriebenen Grafikerweiterungen zurückgreifen. Ab Kapitel 6.3 werden noch einige allgemeine Basic-Ergänzungen angefügt. Zur Benutzung dieser Basic-Erweiterungen ist es im Prinzip nicht notwendig die Maschinen-Programme zu verstehen. In diesen Kapiteln braucht man sich lediglich anzusehen, wie diese Befehle funktionieren, um sie verwenden zu können.

Im letzten Kapitel werden noch einige allgemeine Tips und Tricks für den Commodore 64 vorgestellt, so z.B. die Übernahme von Programmen der Serie 2000, sowie das Speichern von Programmen auf der Serie 8000, um mit den dort gegebenen Hilfsmitteln besser Programme editieren zu können. Weitere nützliche Hinweise folgen.

Im Anhang werden noch einige ergänzende Tabellen dargestellt, und als Abschluß bringen wir noch eine kurze Übersicht, über die Thematik der nächsten Bände.

Das COMMODORE 64 - BUCH

Band 1 : Ein Leitfaden für den Erstanwender

Inhaltsverzeichnis

1

Programme nicht nur für den Commodore 64

1. Programme - nicht nur für den Commodore 64

Um auch den Anfängern, die erst ein wenig Basic anhand eines Basic-Lehrbuches gelernt haben, den Einstieg in den Commodore 64 zu ermöglichen, wurde extra dieses erste Kapitel geschrieben. Mit ihm werden solche Programme behandelt, für die eigentlich die Computer entwickelt wurden: Mathematische Probleme. Mathematische Programme bilden auch ein gutes Beispiel für die diversen Tätigkeiten des Computers: Einlesen von Daten / Verarbeiten von Daten / Ausgabe der verarbeitenten Daten. Sie sind daher sehr gut zur Einführung in die Arbeitsweise eines Computer geeignet. Das einzige was in diesen Programmen speziell für den Commodore 64 vorhanden ist, sind die Sonderzeichen für die Farbgebung. Anhand dieser Programme kann man auch schon seine Lieblingsfarbkombination zusammensuchen, indem die jeweils beschriebenen Farbbefehle (Sonderzeichen in einer Bildschirmausgabe / PRINT-Befehl) den eigenen Wünschen angepasst werden.

1.1 MAMULT - Matrizenmultiplikation

Übungsziel:

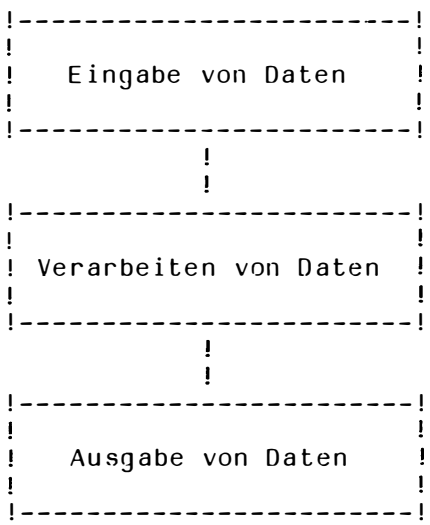
- INPUT-Befehl
- PRINT-Befehl
- einfache mathematische Berechnungen
- IF-Abfragen
- Dimensionierung von Feldern

```
100 POKE53280,6
110 POKE53281,6
120 INPUT"■1.MATRIX - ZEILEN";Z1
130 INPUT"■■.
140 PRINT"■■■SOLL DIE MATRIX MIT SICH SELBST
150 GETA$:IFA$=""THEN150          MULTIPLIZIERT WERDEN";
160 PRINTA$
170 IFA$="J"THEN250
180 IFA$="N"THEN200
190 GOTO150
```

```
200 INPUT"■■2.MATRIX - ZEILEN";Z2
210 INPUT"■■2.MATRIX - SPALTEN";S2
220 IFS2<>Z1THEN270
230 IFZ2<>S1THEN270
240 GOTO1000
250 IFZ1<>S1THEN270
260 GOTO2000
270 PRINT"■■WIE WOLLEN SIE DAS MACHEN?????":END
1000 DIMA(Z1,S1):DIMB(Z2,S2)
1010 PRINT"■■"
1020 FORI=1TOZ1
1030 FORJ=1TOS1
1040 PRINT"■A (";I;",";J;")";
1050 INPUTA(I,J)
1060 NEXT
1070 NEXT
1080 PRINT"■"
1090 FORI=1TOZ2
1100 FORJ=1TOS2
1110 PRINT"B (";I;",";J;")";
1120 INPUTB(I,J)
1130 NEXT
1140 NEXT
1150 DIMC(Z1,S2)
1160 FORI=1TOZ1
1170 FORJ=1TOS2
1180 FORK=1TOS1
1190 C(I,J)=C(I,J)+A(I,K)*B(K,J)
1200 NEXTK
12
1220 NEXTI
1230 PRINT"■"
1240 POKE53281,7
1250 PRINT"■ERGEBNIS : ("Z1" * "S2"-MATRIX )■■"
1260 FORI=1TOZ1
1270 PRINT
1280 FORJ=1TOS2
1290 PRINTC(I,J),
1300 NEXTJ
1310 NEXTI
1320 PRINT"■■■"
1330 END
2000 DIMA(Z1,Z1)
2010 FORI=1TOZ1
2020 FORJ=1TOZ1
2030 PRINT"■■■A (";I;",";J;")";
2040 INPUTA(I,J)
2050 NEXTJ
2060 NEXTI
2070 DIMC(Z1,Z1)
2080 FORI=1TOZ1
```

```
2090 FORJ=1TOZ1
2100 FORK=1TOZ1
2110 C(I,J)=C(I,J)+A(I,K)*A(K,J)
2120 NEXTK
2130 NEXTJ
2140 NEXTI
2150 PRINT"■"
2160 POKE53281,7
2170 POKE53280,7
2180 PRINT"■ERGEBNIS : ("Z1" * "Z1"-MATRIX )■■■"
2190 FORI=1TOZ1
2200 PRINT
2210 FORJ=1TOZ1
2220 PRINTC(I,J);
2230 NEXTJ
2240 NEXTI
2250 PRINT"■■■
2260 END
```

Ein in der Mathematik immer wiederkehrendes Problem ist die Multiplikation zweier Matrizen. Mit dem in diesem Kapitel beschriebenen Programm, wollen wir das Prinzip eines jeden Computereinsatzes verdeutlichen:



In diesem einfachen Falle hier wollen wir das Einlesen über die Tastatur nehmen und die Ausgabe auf dem Bildschirm. In größeren Programmen wird auch immer wieder dieses Schema verfolgt, nur werden die einzelnen Teile immer wieder ineinander geschachtelt und die Ausgabe-/ Eingabe-Medien können auch z.B. externe Speicher wie Kassettenrecorder oder Floppylaufwerke oder Plattengeräte sein.

In den beiden ersten Zeilen wird nur die Hintergrundfarbe des Bildschirms und der Rahmen mit der Farbe blau belegt. Als Schrift wurde schwarz gewählt. Bereits ab Zeile 120 werden Daten eingelesen. Bevor jedoch die Einträge einer Matrix eingelesen werden, muß man zuerst dem Computer sagen, wieviele Zeilen und Spalten die Matrix haben soll (Z1, S1). Auch hier wieder ein Hinweis: Vergeben Sie für die verwendeten Variablen möglichst sogenannte mnemotechnische Bezeichnungen. Z.B. die Variable Z1 deutet auf 'Zeilenzahl 1. Matrix' und die Variable S1 auf 'Spaltenzahl 1. Matrix' hin.

Dann kommt schon eine anwenderfreundliche Eingabe, in dem der Rechner fragt ob die Matrix mit sich selbst multipliziert werden soll. In diesem Falle können die Werte der 2. Matrix aus den Variablen für die Werte der 1. Matrix herausgeholt werden, und die erneute Eingabe der gleichen Matrizenwerte entfällt.

In den Zeilen 150 bis 190 ist eine einfache Tastaturabfrage auf die Buchstaben 'J' und 'N'. Dabei wird zunächst ein Zeichen von der Tastatur eingelesen (GET A$). Im weiteren Verlauf der Zeile 150 wird abgefragt ob ein Zeichen an der Tastatur gedrückt wurde, wenn nicht, wird die Zeile 150 als ständige Warteschleife durchlaufen bis ein Zeichen über die Tastatur eingegeben wird. Dieses Zeichen wird in Zeile 160 am Bildschirm ausgegeben und in den Zeilen 170 und 180 findet eine Plausibilitätsprüfung statt. Gleichzeitig mit der Plausibilitätsprüfung wird entweder die Eingabe der zweiten Matrix übersprungen oder nicht. Wurde ein von 'J' oder 'N' verschiedenes Zeichen eingegeben, so springt der Rechner wieder zur Einleseschleife.

Als nächste Daten werden die Zeilen und Spalten der 2. Matrix eingelesen. Auch hier erfolgt anschließend wieder eine Plausibilitätsprüfung, da bei einer Matrizenmultiplikation die Anzahl der Zeilen in der ersten Matrix mit der Anzahl der Spalten in der zweiten Matrix übereinstimmen muß, und analog auch die Anzahl der Zeilen in der zweiten Matrix mit der Anzahl der Spalten in der Ersten. Ist eine Matrizenmultiplikation mit den eingebenen Zeilen- und Spaltenzahlen nicht möglich, so bricht das Programm mit einer Fehlermeldung ab. Zeile 250 beinhaltet die Plau-

sibilitätsprüfung für die Matrizenmultiplikation mit derselben Matrix. Hier muß die Anzahl der Zeilen mit der Anzahl der Spalten übereinstimmen.

Die Zeilen 1000 bis 1330 beinhalten das Programm zum Berechnen des Multiplikationsergebnisses und ab 2000 wird eine Matrix mit sich selbst multipliziert.

Mit den im ersten Teil erhaltenen Variablen für Zeilen- und Spaltenlängen der diversen Matrizen werden in Zeile 1000 im Rechner die beiden Matrizen dimensioniert. Dann wird die Farbe für die weitere Eingabe umgeschaltet. Die Zeilen 1020 bis 1070 dienen zur Eingabe der Matrizenwerte der ersten Matrix, wobei die Zeile und Spalte des jeweils einzugebenden Elementes am Rechner vorher durch einen PRINT-Befehl ausgegeben werden. Die eingebenen Daten werden sofort in die dimensionierte Matrix übertragen. Die Zeilen 1090 bis 1140 dienen analog zum Einlesen der Werte der zweiten Matrix.

In Zeile 1150 wird die Matrix C(,) für die Ausgabedaten dimensioniert und in den Zeilen 1160 bis 1220 errechnet. Ab Zeile 1230 erfolgt die Ausgabe der Ergebnismatrix bis Zeile 1320. Bei der Berechnung sind drei FOR...NEXT-Schleifen erforderlich, da die Matrizenberechnung nach der allgemein bekannten Formel:

$$C(I,J) = \sum_{K=1}^{N} A(I,K) * B(K,J)$$

ausgeführt wird. Zwei Schleifen laufen jeweils durch die Zeilen bzw. Spalten und in der dritten Schleife werden die schon errechneten Werte aufsummiert. Das Ergebnis wird auch zeilen- und spaltengerecht ausgegeben. Dazu ist unter anderem der PRINT-Befehl in Zeile 1270 gedacht, der nach jeder Zeile ein Zeilenvorschub am Bildschirm bewirkt.

Analog funktioniert das Multiplizieren zweier gleicher Matrizen. Eine Spaltenvariable ist eigentlich nicht mehr nötig, da die Anzahl der Zeilen und Spalten gleich sein muß (sogenannte quadratische Matrizen). Deshalb wurde für alle Rechenvorgänge die Variable Z1 herangezogen.

MAMULT **100 - 2260**

Variablen:

Name	Typ	Bereich	Bedeutung
I	H	1...Z1/Z2	Laufvariable
J	H	1...S1/S2	Laufvariable
K	H	1...S1	Laufvariable
S1	G	Integer	Zahl der Spalten in der ersten Matrix
S2	G	Integer	Zahl der Spalten in der zweiten Matrix
Z1	G	Integer	Zahl der Zeilen in der ersten Matrix
Z2	G	Integer	Zahl der Zeilen in der zweiten Matrix

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
A	Z1,S1	G	Dezimalzahlen	Werte der ersten Matrix
B	Z2,S2	G	Dezimalzahlen	Werte der zweiten Matrix
C	Z1,S2/ Z1,Z1	G	Dezimalzahlen	Werte der Ergebnismatrix

Unterprogrammaufrufe :

in	nach	Zweck
keine!		nur GOTO's

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
220	END	S2 NE Z1	Matrizen nicht multiplizierbar
230	END	Z2 NE S1	Matrizen nicht multiplizierbar
250	END	Z1 NE S1	Matrizen nicht multiplizierbar (keine quadratische Matrix)

1.2 GEO - Geometrische Berechnungen

Übungsziel:

- Programmentwicklung
- Zeilennummernvergabe
- hierarchische Menütechnik
- Farbausgabe
- Durchführung einfacher mathematischer Berechnungen
- Problemanalyse

```
10 POKE53280,11
20 POKE53281,11
100 PRINT"■■WO SIND BERECHNUNGEN DURCHZUFUEHREN ?■"
110 PRINTTAB(10);"   KREIS      --> K■"
120 PRINTTAB(10);"   QUADRAT    --> Q■"
130 PRINTTAB(10);"   RECHTECK   --> R■"
140 PRINTTAB(10.     DREIECK    --> D■"
150 PRINTTAB(10);"   WUERFEL    --> W■"
160 PRINTTAB(10);"   QUADER     --> A■"
170 PRINTTAB(10);"   PRISMA     --> P■"
180 PRINTTAB(10);"   TETRAEDER  --> T■"
190 PRINTTAB(10);"   ZYLINDER   --> Z■"
200 PRINTTAB(10);"   KEGEL      --> E■"
210 PRINTTAB(10);"   KUGEL      --> U■"
230 GE!A$
240 IFA$="K"THEN1000
250 IFA$="Q"THEN1500
260 IFA$="R"THEN2000
270 IFA$="D"THEN2500
```

```
280 IFA$="W"THEN3000
290 IFA$="A"THEN3500
300 IFA$="P"THEN4000
310 IFA$="T"THEN4500
320 IFA$="Z"THEN5000
330 IFA$="E"THEN5500
340 IFA$="U"THEN6000
350 GOTO230
1000 GOSUB9000
1010 PRINT"■RADIUS";TAB(15);"■UMFANG              --> 1■"
1020 PRINT"■UMFANG";TAB(15);"■RADIUS              --> 2■"
1200 GETA$:IFA$=""THEN1200
1210 A=VAL(A$)
1220 IFA<1ORA>2THEN1200
1230 ON A GOTO 10000,10100
1500 GOSUB9000
1510 PRINT"■KANTE";TAB(15);"■DIAG.,UMF.,FLAE. --> 1■"
1520 PRINT"■UMFANG"
1530
1540 PRINT"■FLAECHE";TAB(15);"■DIAG.,KANT .
1600 GETA$:IFA$=""THEN1600
1610 A=VAL
1620 IFA<1ORA>4THEN1600
1630 ON A GOTO 15000,15200,15400,15600
2000 GOSUB9000
2010 PRINT"■KANTEN                          -->1"
2020 PRINT"■KANTEN";TAB(15);"■UMFANG        -->1"
2030 PRINT"■KANTEN";TAB(15);"■FLAECHE       -->1"
2040 PRINT"■KANTE,FLAECHE";TAB(15)                  -->2"
2050 PRINT"■KANTE,FLAECHE";TAB(15);"■DIAGON         -->2"
2060 PRINT"■KANTE,FLAECHE";TAB(15);"■UMFANG         -->2"
2070 PRINT"■KANTE,DIAG.   ";TAB(15);"■2.KANTE       -->3"
2080 PRINT"■KANTE,DIAG.   ";TAB(                    -->3"
2090 PRINT"■KANTE,DIAG.   ";TAB(15);"■FLAECHE       -->3"
2100 PRINT"■KANTE                                   -->4"
2110 PRINT"■KANTE,                                  -->4"
2120 PRINT"■KANTE,UMFANG ";TAB(15);"■FLAECHE        -->4"
2200 GETA$:IFA$=""THEN2200
2210 A=VAL(A$)
2220 IFA<1ORA>4THEN2200
2230 ON A GOTO 20000,20200,20400,20600
2500 GOSUB9000
2510 PRINT"■KANTEN";TAB(15);"■FLAECHE          -->1■■"
2520 PRINT"■KANTE,HOEHE  ";                            -->2"
2600 GETA$:IFA$=""THEN2600
2610 A=VAL(A$)
2620 IFA<1ORA>2THEN2600
```

```
2630 ON A GOTO 25000,25200
3000 GOSUB9000
3010 PRINT"■KANTEN";TAB(15);"■VOLUMEN          -->1"
3020 PRINT"■KANTEN";TAB(15);"■OBERFLAECHE      -->1"
3030 PRINT"■KANTEN";TAB(15);"■DIAGONALE        -->1"
3040 PRINT"■OBERFLAECHE";TAB(15);"■KANTEN          -->2"
3050 PRINT"■OBERFLAECHE";TAB(15);"■DIAGONALE       -->2"
3060 PRINT"■OBERFLAECHE"                           -->3"
3070 PRINT"■DIAGONALE";TAB(15);"■KANTEN            -->3"
3080 PRINT"■DIAGONALE";TAB(15);"■VOLUMEN           -->3"
3090 PRINT"■DIAGONALE";TAB(15);"■OBERFLAECHE       -->3"
3200 GET A$:IF A$="" THEN 3200
3210 A=VAL(A$)
3220 IF A<1 OR A>3 THEN 3200
3230 ON A GOTO 30000,30200,30400
3500 GOSUB9000
3510 PRINT"■3 KANTEN";TAB(15);"■VOLUMEN          -->1"
3520 PRINT"■3 KANTEN";TAB(15);"■OBERFLAECHE      -->1"
3530 PRINT"■3 KANTEN";TAB(15);"■DIAGONALE        -->1"
3540 PRINT"■2 KANTEN,DIAG.";TAB(15);"■OBERFLAEC      -->2"
3550 PRINT"■2 KANTEN,DIAG.";TAB(15);"■3. KANTE       -->2"
3560 PRINT"■2 KANTEN,DIAG.";TAB(15);"■VOLUMEN        -->2"
3570 PRINT"■2 KANTEN,OBFL.";TAB(15);"■VOLUMEN        -->3"
3580 PRINT"■2 KANTEN,OBFL.";TAB(15);"■DIAGONALE      -->3"
3590 PRINT"■2 KANTEN,OBFL.";TAB(15);"■3. KANTE       -->3"
3600 PRINT"■2 KANTEN,VOL.";TAB(15);"■3. KANTE        -->4"
3610 PRINT"■2 KANTEN,VOL.";TAB(15);"■OBERFLAECHE     -->4"
3620 PRINT"■2 KANTEN,VOL.";TAB(15);"■DIAGONALE       -->4"
3700 GET A$:IF A$="" THEN 3700
3710 A=VAL(A$)
3720 IF A<1 OR A>4 THEN 3700
3730 ON A GOTO 35000,35200,35400,35600
4000 GOSUB62000
4010 GOTO61000
4500 GOSUB62000
4510 GOTO61000
5000 GOSUB62000
5010 GOTO61000
5500 GOS
5510 GOTO61000
6000 GOSUB62000
6010 GOTO61000
9999 END
9000 PRINT"■■GEGEBEN          ■GESUCHT"
9010 PRINT"■********          ■********■■"
9020 RETURN
10000 INPUT"■■RADIUS";R
10010 U=2*π*R
10020 I=π*R↑2
10030 PRINT"■■■UMFANG  :",U
10040 PRINT"■■■INHALT  :",I
```

```
10050 GOTO61000
10100 INPUT"■UMFANG";U
10110 R=U/(2*π)
10120 D=U/π
10130 PRINT"■■RADIUS :",,R
10140 PRINT"■■DURCHMESSER :",D
10150 GOTO61000
10200 INPUT"■UMFANG"
10210 D=U/π
10220 PRINT"■■DURCHMESSER :",D
10230 GOTO61000
10300 INPUT"■RADIUS ";R
10310 I=π*R↑2
10320 PRINT"■■INHALT :",I
10330 GOTO61000
15000 INPUT"■KANTENLAENGE";K
15010 D=K*SQR(2)
15020 U=4*K
15030 A=K↑2
15040 PRINT"■■DIAGONALE:",D
1
15060 PRINT"■■FLAECHE  :",A
15070 GOTO 61000
15200 INPUT"■UMFANG";U
15210 K=U/4
15220 D=K*SQR(2)
15230 A=K↑2
15240 PRINT"■■KANTENLAENGE :",K
15250 PRINT"■■DIAGONALE    :",D
15260 PRINT"■■FLAECHE      :",A
15270 GOTO61000
15400 INPUT"■DIAGONALE";D
15410 K=D/SQR(2)
15420 U=4*K
15430 A=K↑2
15440 PRINT"■■KANTENLAENGE :",K
15450 PRINT"■■UMFANG       :",U
15460 PRINT"■■FLAECHE      :",A
15470 GOTO61000
15600 INPUT"■FLAECHE";A
15610 K=SQR(A)
15620 U=4*K
15630 D=K*SQR(2)
15640 PRINT"■■KANTENLAENGE :",K
15650 PRINT"■■UMFANG       :",U
15660 PRINT"■■DIAGONALE    :",D
15670 GOTO61000
20000 INPUT"■KANTE A";KA
20010 INPUT"■■■■■■■■■■■■■■■■■■■■■■KANTE B";KB
20020 D=SQR(KA↑2+KB↑2)
20030 U=2*(KA+KB)
```

```
20040 A=KA*KB
20050 PRINT"■■■DIAGONALE :",D
20060 PRINT"■■UMFANG    :",U
20070 PRINT"■■FLAECHE   :",A
20080 GOTO61000
20200 INPUT"■■KANTE   A ";KA
20210 INPUT"■FLAECHE F ";F
20220 B=F/KA
20230 D=SQR(KA↑2+B↑2)
20240 U=2*(KA+B)
20250 PRINT"■■■ANDERE KANTE :",B
20260 PRINT"■■DIAGONALE    :",D
20270 PRINT"■■UMFANG       :",U
20280 GOTO61000
20400 INPUT"■■KANTE A   ";A
20410 INPUT"■DIAGONALE ";D
20420 B=SQR(D↑2-A↑2)
20430 U=2*(A+B)
20440 A=A*B
20450 PRINT"■■■ANDERE KANTE :",B
20460 PRINT"■■UMFANG        :",U
20470 PRINT"■■FLAECHE       :",A
20480 GOTO61000
20600 INPUT"■■KANTE A";A
20610 INPUT"■UMFANG ";U
20620 B=U/2-A
20630 D=SQR(A↑2+B↑2)
20640 A=A*B
20650 PRINT"■■■ANDERE KANTE :",B
20660 PRINT"■■DIAGONALE    :",D
20665 PRINT"■■FLAECHE      :",A
20670 GOTO61000
25000 INPUT"■■KANTE A";KA
25010 INPUT"■KANTE B";KB
25020 INPUT"■KANTE C";KC
25030 S=.5*(KA+KB+KC)
25040 FL=SQR(S*(S-KA)*(S-KB)*(S-KC))
25050 PRINT"■■■FLAECHE :",FL
25060 GOTO61000
25200 INPUT"■■KANTE A";KA
25210 INPUT"■HOEHE A";HA
25220 FL=KA*HA/2
25230 PRINT"■■■FLAECHE :",FL
25240 GOTO61000
30000 INPUT"■■KANTEN";KA
30010 V=KA↑3
30020 D=KA*SQR(3)
30030 O=6*KA↑2
30040 PRINT"■■■VOLUMEN :",,V
30050 PRINT"■■DIAGONALE :",D
30060 PRINT"■■OBERFLAECHE :",O
```

```
30070 GOTO61000
30200 INPUT"■■OBERFLAECHE";O
30210 KA=SQR(O/6)
30220 D=KA*SQR(3)
30230 V=6*KA↑2
30240 PRINT"■■KANTEN :",,KA
30250 PRINT"■■DIAGONALE :",D
30260 PRINT"■■VOLUMEN :",,V
30270 GOTO61000
30400 INPUT"■■DIAGONALE";D
30410 KA=D/SQR(3)
30420 O=KA6*KA↑2
30430 V=6*KA↑2
30440 PRINT"■■KANTEN :",,KA
30450 PRINT"■■OBERFLAECHE :",O
30460 PRINT"■■VOLUMEN :",,V
30470 GOTO61000
35000 INPUT"■■KANTE A";KA
35010 INPUT"KANTE B";KB
35020 INPUT"KANTE C";KC
35030 V=KA*KB*KC
35040 D=SQR(KA↑2+KB↑2+KC↑2)
35050 O=2*(KA*KB+KA*KC+KB*KC)
35060 PRINT"■■VOLUMEN :",,V
35070 PRINT"■■DIAGONALE :",D
35080 PRINT"■■OBERFLAECHE :",O
35090 GOTO61000
35200 REM ****** UEBUNG FUR DEN LESER ******
35210 REM FORMEL FUER 3. KANTE :
35220 KC = D↑2 - KA↑2 - KB↑2
35400 REM ****** UEBUNG FUR DEN LESER ******
35410 REM FORMEL FUER 3. KANTE :
35420 REM KC = ( O/2 - KA*KB ) / (A+B)
35600 REM ****** UEBUNG FUR DEN LESER ******
35610 REM FORMEL FUER 3. KANTE :
35620 REM KC = V / ( KA * KC )
37000 GOSUB 62000:GOTO61000
40000 REM FORMEL FUER PRISMA
40010 REM V = G * H
40020 REM O = 2 * G + M
45000 REM FORMEL FUER TETRAEDER
45010 REM V = 1/12 * A↑3 * SQR(2)
45020 REM H = A/3 * SQR(6)
45030 REM O = A↑2 * SQR(3)
50000 REM FORMEL FUER ZY$YDER
50010 REM V = G * H
55000 REM FORMEL FUER KEGEL
55010 REM V = 1/3 * G * H
60000 REM FORMEL FUER KUGEL
60010 REM V = 4/3 * π * R↑3
60020 REM O = 4 * π * R↑2
```

```
61000 PRINT"■■■■■■■■■■■■■WEITER ?  --> TASTE DRUECKEN"
61010 GETA$:IFA$=""THEN61010
61020 GOTO100
62000 PRINT"■BITTE VERSUCHEN SIE SICH SELBST
62010 PRINT"■■IN DER PROGRAMMIERUNG !!!!!
62020 PRINT"■■■FORMELN IN REM-ANWEISUNGEN
62030 PRINT"■■■IN DEN VORGESEHENEN ZEILEN
62040 RETURN
63000 SAVE"@:GEOMETRIE",8
63010 OPEN1,8,15
63020 INPUT#1,A,B$,C,D
63030 PRINTA,B$,C,D
63040 END
```

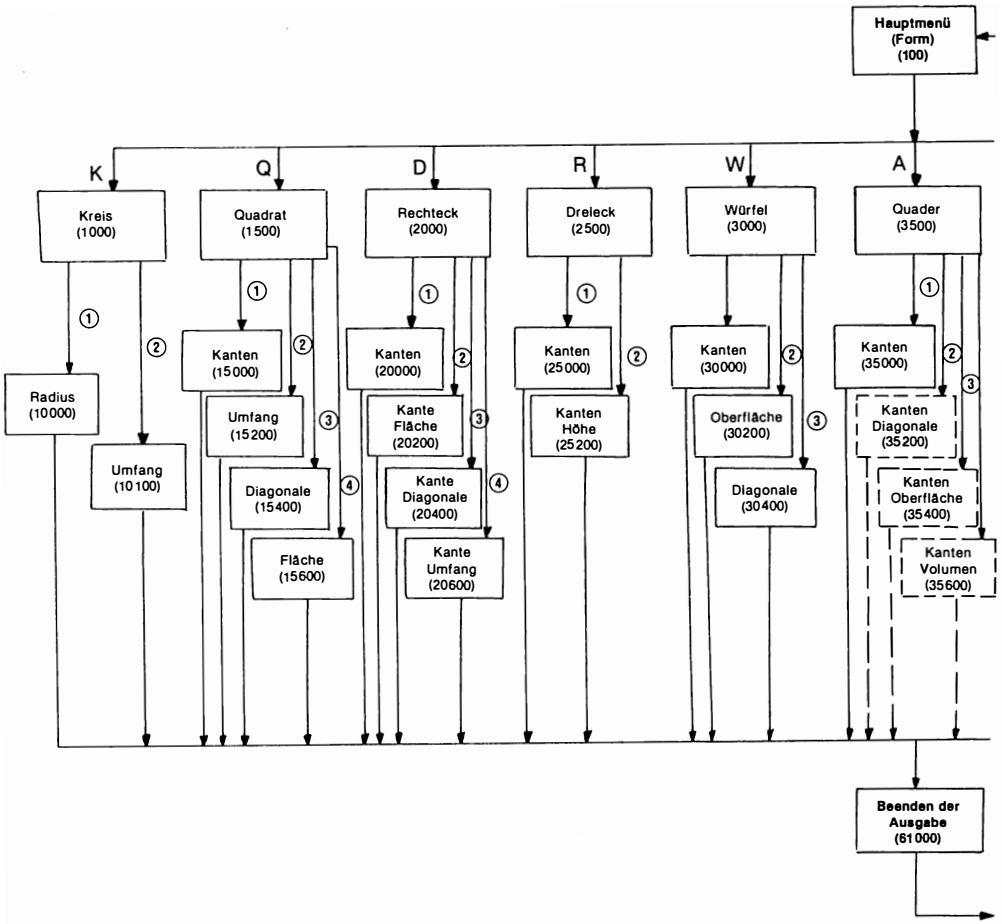
Ähnlich wie in dem Programm zur Berechnung von Matrizen (MAMULT) wollen wir hier mathematische Berechnungen durchführen. Um nicht für jede Berechnung ein eigenes Programm schreiben zu müssen, werden diverse Berechnungen in einem Programm zusammengefaßt, dem ein Menü vorangestellt wird. Wir wollen mathematische Berechnungen an verschiedenen geometrischen Formen vornehmen. Dazu wollen wir erst einmal festlegen, an welchen Formen wir Berechnungen durchführen wollen. Für das aufgezeigte Programm haben wir festgelegt:

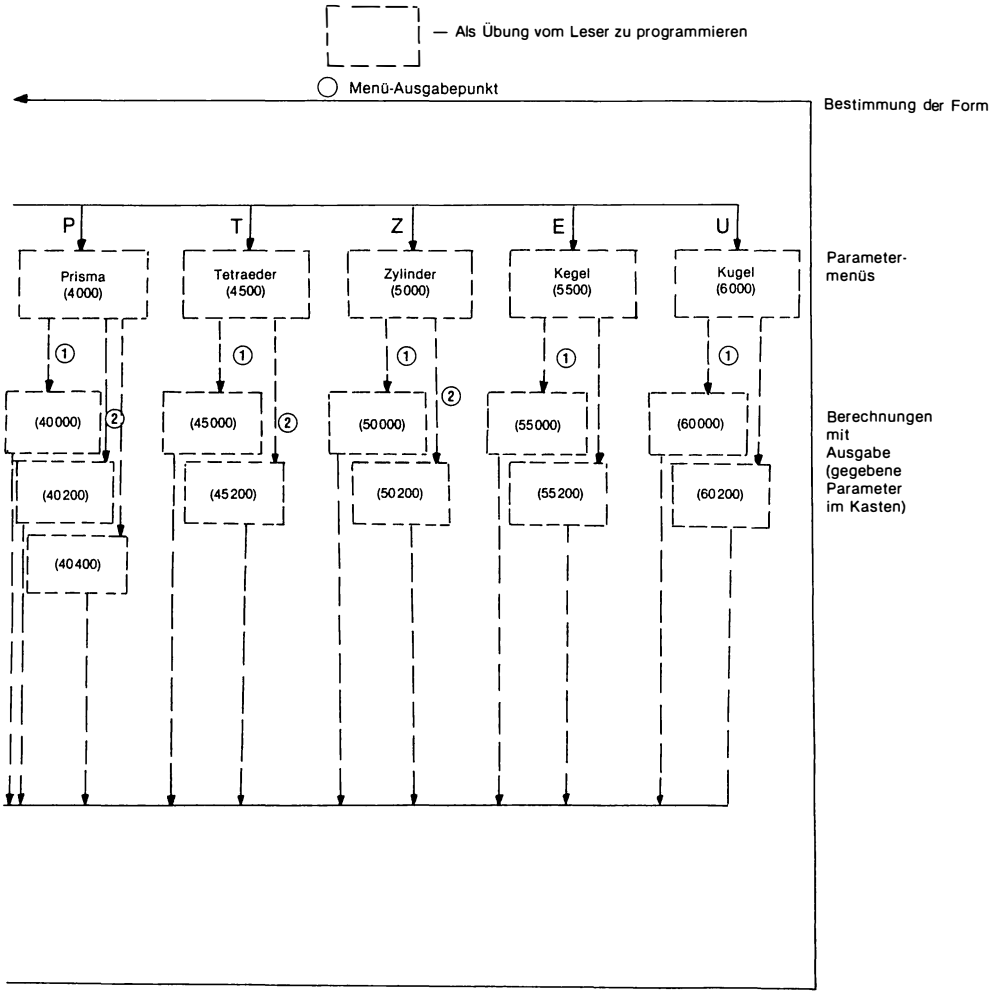
- Kreis
- Quadrat
- Rechteck
- Würfel
- Dreieck
- Quader
- Prisma
- Tetraeter
- Zylinder
- Kegel
- Kugel

Ob Sie das Programm nun Top-Down oder Bottom-Up entwickeln (vgl. auch Basic ohne Probleme Band 3 - Programmentwicklung und Datenverwaltung), bleibt Ihnen überlassen. Wir wollen hier den Fall der Top-Down-Programmierung darstellen, wo das Programm vom Gerüst bis in die Details immer weiter verfeinert wird.

Dazu brauchen wir zunächst einmal das Menü, was zu den Berechnungen der einzelnen geometrischen Figuren verzweigen wird. Dieses Menü zeigt die Zeilen 100 bis 350.
Vorangestellt sind lediglich noch zwei Befehle um dem Commodore 64 als Rahmen- und Hintergrundfarbe jeweils dunkelgrau zu geben, damit andere Farben besser zur Geltung kom-

Grobes Ablaufpogramm »Geometrie«. Die Zahlen in Klammern geben den Beginn des jeweiligen Programmstückes an, die Zeichen vor den Kästen den anzuwählenden Menüpunkt.




— Als Übung vom Leser zu programmieren
Menü-Ausgabepunkt
Bestimmung der Form
P
T
Z
E
U
Prisma (4000)
Tetraeder (4500)
Zylinder (5000)
Kegel (5500)
Kugel (6000)
Parameter-menüs
(40000)
(45000)
(50000)
(55000)
(60000)
(40200)
(45200)
(50200)
(55200)
(60200)
(40400)
Berechnungen mit Ausgabe (gegebene Parameter im Kasten)

men. Nehmen Sie diese Poke-Befehle zunächst einmal als gegeben hin.

Das Menü ist unterteilt in die Bildschirmanzeige (Zeile 100 bis 210) und den Verzweigungsteil (Zeile 230 bis 350), wobei in Zeile 230 ein Zeichen von der Tastatur eingelesen wird und ab Zeile 240 in Abhängigkeit des eingelesenen Zeichens verzweigt wird. Ist ein Zeichen eingegeben worden, welches nicht zum Programmverteiler gehört, so wird ein weiteres Zeichen von der Tastatur eingelesen.

Der Übersichtlichkeit halber wollen wir die geometrischen Berechnungen jeweils bei Zeilennummern ab 1000 anfangen lassen, wobei für jede weitere geometrische Figur die erste Zeile um 500 höher liegen soll als bei der vorhergehenden Figur. Dies gestattet uns, wie wir später sehen werden, den gesamten möglichen Zeilenbereich in Basic auszunutzen. Die Vergabe der Zeilennummern ist möglichst schon im voraus festzulegen, um nicht später nach dem Motto verfahren zu müssen 'machen wir halt ein GOTO in einen neuen Speicherbereich', wenn der Zeilenbereich, der vorgesehen wurde, erschöpft ist. Sogenannte 'organisch gewachsene' Programme stellen erhebliche Probleme bei der Programmpflege und Wartung dar.

Bei den im Hauptmenü angesprungenen Zeilennummern soll jeweils ein Untermenü stehen, mit dem abgefragt werden soll, welche Parameter der Figur gegeben sind und welche berechnet werden sollen. Aus mnemotechnischen Gründen sollen die Menüs ab der Zeilennummer stehen, die im Hauptmenü angegeben ist, und die eigentlichen Berechnungen jeweils in einer Zeilennummer beginnen, die eine 0 mehr hat als die erste Zeile des Untermenüs. So steht z.B. das Untermenü zum Abfragen der Parameter beim Kreis ab Zeile 1000 und die Berechnungen ab Zeile 10000. Desgleichen beginnt das Untermenü zu den Berechnungen am Quadrat in Zeile 1500 und die Berechnungen selbst in 15000. Somit ist dann der Speicherbereich bis Zeile 60000 ff. ausgelastet.

Um Speicherplatz und Tipp-Arbeit zu sparen, wollen wir die Überschrift für die gegebenen und gesuchten Parameter in ein Unterprogramm zusammenfassen, das vor jedem Untermenü aufgerufen wird (Zeilen ab 9000).

Die Untermenüs sind alle analog aufgebaut. Jedem Hauptmenü werden zunächst die Menüpunkte am Bildschirm ausgegeben und anschließend über die Abfrage eines Zeichens von der Tastatur in die verschiedenen Berechnungen verzweigt. Ein Unterschied zum Haupmenü besteht allerdings: Im Hauptmenü wird nach einzelnen Zeichen (Buchstaben) verzweigt. In den Untermenüs haben wir ein anderes Verfahren verwendet. Die

Menüpunkte werden über Ziffern angewählt, sodaß der Befehl 'ON A GOTO' verwendet werden kann. Dadurch wird Zeilen- und Speicherplatz gespart. Die Zeilen analog zu Zeile 1220 beinhalten jeweils eine Plausibilitätsprüfung. Dadurch wird die Tastatur softwaremäßig für alle anderen Zeichen, als die im Menü vorgegebenen, gesperrt.

Bei den eigentlichen Berechnungen (ab Zeile 10000) geschieht im Prinzip das Gleiche wie in dem Programm MAMULT. Zunächst werden die Daten, die Grundlage der Berechnung sind, am Bildschirm eingelesen, anschließend die Berechnung durchgeführt, das Ergebnis ausgedruckt, und - anstatt das Programm zu beenden - ein Programmstück aufgerufen, was auf das Drücken irgendeiner Taste wartet. Dadurch werden die Berechnungen solange am Bildschirm angezeigt, wie es der Benutzer will. Durch Drücken einer beliebigen Taste wird wieder zum Hauptmenü übergegangen.

Im Prinzip sind hier viele kleine Programme zu einem großen zusammengefasst. Die Berechnungen wurden absichtlich in dieser einfachen Form gehalten, um über das Programm trotz der Vielfalt der 'kleinen Programme' einen Überblick zu erhalten. An diesem Beispiel wird leicht deutlich, daß auch geschachtelte Menüs nicht schwieriger zu handhaben sind als einfache Menüs, man muß sich vorher nur über die Verteilung der Zeilennummern Gedanken machen.

Änderungen und Ergänzungen

Wie auf dem nebenstehenden Flußdiagramm schon durch die gestrichelten Linien deutlich gemacht und im Programmlisting an den REM-Anweisungen ersichtlich, sollen Sie selbst einige von diesen Berechnungen und Untermenüs eintippen. Gehen Sie aber dabei bitte sehr sorgfältig vor. Laufen die Programmstücke, die Sie selbst eingetippt haben nicht auf Anhieb, so gehen Sie logisch bei der Fehlersuche vor. Die am leichtesten zu erkennenden Fehler sind die 'syntax error'-Fehler, weil der Computer sie anzeigt. Achten Sie dann auf die richtige Schreibweise von Basic-Schlüsselwörtern, gleiche Anzahl von geöffneten und geschlossenen Klammern und Variablennamen, die keine Schlüsselwörter sein dürfen.

Die Zeichen, die in Reverse im Listing dargestellt sind, sind Steuerzeichen für den Cursor und die Farbgebung. Näheres zu den Farben ist aus der Tabelle im Anhang ersichtlich. Vom Programm vorgewählt sind folgende Farben:

weiß - für das Hauptmenü
grün - für die gegebenen Parameter

hellrot	-	für die gesuchten Parameter (das normale rot ist auf dem Hintergrund nicht so gut lesbar)
gelb	-	für die eingegebenen Daten
blau	-	für die ausgegebenen Daten

Zum Beispiel ab Zeile 2010 befindet sich hinter dem '"' des PRINT-Befehls ein Reverse-Pfeil nach oben, der die Ausgabefarbe auf 'grün' umschaltet und vor dem 2. Text ein Reverse-Eckzeichen, das Ausgabe auf hellrot umschaltet. Diese Farben müssen in jeder Zeile umgeschaltet werden.

Bei der Ausgabe von Daten wurde die Farbumschaltung je Zeile zum Teil auch noch durchgeführt (vergl. z.B. Zeile 15040 bis 15060) was im Prinzip jedoch nicht nötig ist. Vergleichen Sie die Ausgabe bei den Berechnungen am Quadrat und die Ausgabe bei den Berechnungen am Rechteck, wo bei der Ausgabe nur noch die erste Ausgabezeile einen Farb-Schaltbefehl bekommen hat.

Versuchen Sie, nicht nur die fehlenden Programmstücke zu ergänzen, sondern auch das Hauptmenü weiter auszubauen mit weiteren geometrischen Formen. Sollte das Hauptmenü größer werden als der Bildschirm Zeilen hat, so lassen Sie einfach das Reverse q am Ende der Ausgabezeile in dem Hauptmenü (Zeile 110 bis 210) weg. Dadurch werden die einzelnen Menüpunkte dichter aneinander gerückt. Eine Übersicht der Cursorsteuerzeichen befindet sich ebenfalls im Anhang.

Sonstiges

Der Befehl TAB(15) bzw. TAB(10) setzt den Cursor in die Spalte 15 bzw. 10 der aktuellen Zeile. Sie sparen sich dadurch die Ausgabe von Blanks (Leerzeichen) oder 'Cursor rechts' und somit Speicherplatz.

Ab Zeile 63000 steht ein kurzes Programm das für Floppy-Benutzer gedacht ist. Mit ihm wird die aktuelle Programmversion auf Diskette gespeichert, wobei der Klammeraffe und der Doppelpunkt vor dem Dateinamen bedeuten, daß eine vorhergehende Datei gleichen Namens zuerst gelöscht werden soll. Die Zeilen 63010 bis 63030 beinhalten das Öffnen einer Datei mit Fehlerkanal (Datei 1, für Gerät Nr. 8, mit Fehlerkanal 15). Dann wird die von der Floppy bereitgestellte Fehlermeldung in die Variablen a,b$,c,d übernommen und anschließend ausgedruckt.

GEO **100 - 62040**

Variablen:

Name	Typ	Bereich	Bedeutung
A	H	0...10	ggf. Zahlwert von A$
A$	H	1 Zeichen	Einlesen eines Zeichens von der Tastatur
D	G	Dezimalzahl	Durchmesser/Diagonale
FL	G	Dezimalzahl	FlÄche
HA	G	Dezimalzahl	HÖhe zur Kante A
I	G	Dezimalzahl	Kreisinhalt
K	G	Dezimalzahl	KantenlÄnge
KA	G	Dezimalzahl	Kante A
KB	G	Dezimalzahl	Kante B
KC	G	Dezimalzahl	Kante C
O	G	Dezimalzahl	OberflÄche
R	G	Dezimalzahl	Radius
U	G	Dezimalzahl	Umfang
V	G	Dezimalzahl	Volumen

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
keine				

Unterprogrammaufrufe :

in	nach	Zweck
1000	9000	BildschirmÜberschrift
1500	9000	BildschirmÜberschrift
2000	9000	BildschirmÜberschrift
2500	9000	BildschirmÜberschrift
3000	9000	BildschirmÜberschrift
3500	9000	BildschirmÜberschrift
4000	62000	Bildschirmmeldung 'Selbstprogrammierung'
4500	62000	Bildschirmmeldung 'Selbstprogrammierung'
5000	62000	Bildschirmmeldung 'Selbstprogrammierung'

5500	62000	Bildschirmmeldung 'Selbstprogrammierung'
6000	62000	Bildschirmmeldung 'Selbstprogrammierung'
----	----	der Rest wird mit 'GOTO' verzweigt

Verzweigungen nach au§en :

in	nach	Bedingung	Bemerkung
keine			

1.3 STATISTIK - Weitere mathematische Berechnungen

Übungsziel:

- statistische Berechnungen
- Verwendung von Hilfsvariablen
- mnemotechnische Zeilennummernvergabe
- Menüs (geschachtelt)
- kombinierter INPUT mit vorherigem PRINT-Befehl
- Duplizieren ähnlicher Zeilen

```
10 POKE53281,11
20 POKE53280,11
30 DIMN(255)
100 PRINT"■■■■■■■■■■■■■■■1 - MITTELWERT / VARIANZ
110 PRINT"■■■■■■■■■■■■    ( STICHPROBE )
120 PRINT"■■■■■■■■■■■■2 - KOMBINATORIK
130 PRINT"■■■■■■■■■■■■3 - BINOMINALKOEFFIZIENT
140 PRINT"■■■■■■■■■■■■4 - BINOMINALVERTEILUNG
300 GETA$
310 IFA$=""THEN300
320 A=VAL(A$)
330 ONAGOTO1000,2000,3000,4000
1000 INPUT"■■■■GROESSE DER STICHPROBE";N
1010 FORI=1TON
1020 PRINT"WERT";I,
1030 INPUTN(I)
1040 NEXT
1050 X=0
1060 FORI=1TON
1070 X=X+N(I)
1080 NEXT
1090 X0=X/N
```

```
1100 PRINT"■■■■■■■■■■■■MITTELWERT  : ";XQ
1110 H1=0
1120 FORI=1TON
1130 H1=H1+(N(I)-XQ)↑2
1140 NEXT
1150 SQ=1/(N-1)*H1
1160 PRINT"■■■■■■■■■■■■VARIANZ     : ";SQ
1170 PRINT"■■■■■■■■■■■■WEITER ?    --> TASTE
1180 GETA$
1190 IFA$=""THEN1180
1200 GOTO100
2000 PRINT"■■■■■■■■■■■■1 - PERMUTATIONEN / FAKULTAET
2010 PRINT"■■■■■■■2 - KOMBINATIONEN OHNE WIEDERHOLUNG
2020 PRINT"■■■■■■3 - KOMBINATIONEN MIT WIEDERHOLUNG
2030 GETA$
2040 IFA$=""THEN2030
2050 A=VAL(A$)
2060 ONAGOTO2100,2200,2600
2100 INPUT"■N";N
2110 FA=1
2120 FORI=1TON
2130 FA=FA*I
2140 NEXT
2150 PRINT"■■■■■■■■■■■■N ! = ";FA
2160 PRINT"■■■■■■     ( = ANZAHL DER PERMUTATIONEN )
2170 GOTO10000
2200 INPUT"■■■■■■■■■■■■N";N
2210 INPUT"■■■■■■■■■■■■K";K
2220 PRINT"■■■■■■■■■■■■MIT BERUECKSICHTIGUNG DER ANORDNUNG
2230 GETA$                                                (J/N)
2240 IFA$="N"THEN2270
2250 IFA$="J"THEN2440
2260 GOTO2230
2270 H1=N
2280 GOSUB11000
2290 H2=FA
2300 H1=N-K
2310 GOSUB11000
2320 H3=FA
2330 H1=K
2340 GOSUB11000
2350 H4=FA
2360 H2=H2/(H4*H3)
2370 PRINT"■■■■■■■ANZAHL DER KOMBINATIONEN VON "
2380 PRINTN;" ELEMENTEN ";K;"-TER ORDNUNG";
2390 PRINT"OHNE"
2400 PRINT"BERUECKSICHTIGUNG DER ANORDNUNG  : "
2410 PRINT
2420 PRINT"■■■■■■■■■■■■■■■■■■";H2
2430 GOTO10000
2440 H1=N
```

```
2450 GOSUB11000
2460 H2=FA
2470 H1=N-K
2480 GOSUB11000
2490 H2=H2/FA
2500 PRINT"■■■■■■ANZAHL DER KOMBINATIONEN VON "
2510 PRINTN;" ELEMENTEN ";K;"-TER ORDNUNG";
2520 PRINT"MIT"
2530 PRINT"BERUECKSICHTIGUNG DER ANORDNUNG  : "
2540 PRINT
2550 PRINT"■■■■■■■■■■■■■■";H2
2560 GOTO10000
2600 INPUT"■■■■■■■■■■■N";N
2610 INPUT"■■■■■■■■■■K";K
2620 PRINT"■■■■■■■■■■MIT BERUECKSICHTIGUNG DER ANORDNUNG
2630 GETA$                                              (J/N)
2640 IFA$="N"THEN2670
2650 IFA$="J"THEN2850
2660 GOTO2630
2670 N=N+K-1
2680 H1=N
2690 GOSUB11000
2700 H2=FA
2710 H1=N-K
2720 GOSUB11000
2730 H3=FA
2740 H1=K
2750 GOSUB11000
2760 H4=FA
2770 H2=H2/(H4*H3)
2780 PRINT"■■■■■■■ANZAHL DER KOMBINATIONEN VON "
2790 PRINTN;" ELEMENTEN ";K;"-TER ORDNUNG";
2800 PRINT"OHNE"
2810 PRINT"BERUECKSICHTIGUNG DER ANORDNUNG  : "
2820 PRINT
2830 PRINT"■■■■■■■■■■■■■■";H2
2840 GOTO10000
2850 H8=N↑K
2860 PRINT"■■■■■■■ANZAHL DER KOMBINATIONEN VON "
2870 PRINTN;" ELEMENTEN ";K;"-TER ORDNUNG";
2880 PRINT"MIT"
2890 PRINT"BERUECKSICHTIGUNG DER ANORDNUNG  : "
2900 PRINT"■■■■■■■■■■■■■■";H8
2910 GOTO10000
3000 INPUT"■■■■■■■■■■■N";N
3010 INPUT"■■■■■■■■■■K";K
3020 H1=N
3030 GOSUB11000
3040 H2=FA
3050 H1=N-K
3060 GOSUB11000
```

```
3070 H3=FA
3080 H1=K
3090 GOSUB11000
3100 H4=FA
3110 H2=H2/(H4*H3)
3120 PRINT"■■■■■■BINOMINALKOEFFIZIENT ";N;" UEBER ";K" :"
3130 PRINT:PRINT:PRINT
3140 PRINT"              ■";H2
3150 GOTO10000
4000 INPUT"■N";N
4010 INPUT"■P";P
4020 Q=1-P
4030 INPUT"■K";X
4040 H1=N
4050 GOSUB11000
4060 H2=FA
4070 H1=N-X
4080 GOSUB11000
4090 H3=FA
4100 H1=K
4110 GOSUB11000
4120 H4=FA
4130 H2=H2/(H4*H3)
4140 F=H2*P↑X*Q↑(N-X)
4150 PRINT"■■■■■■■■■■BINOMINALVERTEILUNG :"
4160 PRINT"■■■              ■";F
4170 PRINT"■■■MITTELWERT  : ";N*P
4180 PRINT"■■■VARIANZ     : ";N*P*Q
4190 GOTO10000
10000 PRINT"■■■■■■■■■■WEITER ?   --> TASTE
10010 GET A$
10020 IF A$=""THEN10010
10030 GOTO100
11000 FA=1
11010 FORI=1TOH1
11020 FA=FA*I
11030 NEXT
11040 RETURN
60000 SAVE"@:STATISTIK",8
```

Wie auch in den beiden vorhergehenden Kapiteln wollen wir uns hier noch den mathematischen Berechnungen widmen. Die Menü-Technik dürfte hinlänglich bekannt sein und soll hier nicht näher erläutert werden. Das vorliegende Programm kann nach belieben ausgebaut werden. In die bisherige Ausbaustufe wurden aufgenommen:

1 - Mittelwert / Varianz einer Stichprobe
2 - Kombinatorik
3 - Binominalkoeffizient
4 - Binominalverteilung

Poissonverteilung und Normalverteilung sowie Regressionsgeraden und Hypothesentest sowie Korrelation können dem Leser als weiteren Übungsstoff dienen.

Mittelwert / Varianz einer Stichprobe

In den Zeilen 1000 bis 1200 wird der Mittelwert einer Stichprobe und ihre Varianz errechnet. Dazu muß zuerst die Größe der Stichprobe erfaßt werden (N) und anschließend die Stichprobenwerte. Dies geschieht indem zunächst am Bildschirm ausgegeben wird 'WERT I' und dann der Wert sofort in das Feld N() übernommen wird. Sicherlich ist es auch möglich, bei einem INPUT-Befehl (siehe Zeile 1000) einen Text miteinzugeben, jedoch läßt sich auf diese Art und Weise keine Variable einbauen. Es wurde deshalb der Umweg über den PRINT-Befehl gewählt, wobei das ';' hinter 'WERT' von Bedeutung ist, sodaß die Anzeige der Nummer des Wertes sofort dahinter erscheint. Durch das ',' hinter dem 'I' wird der eigentliche INPUT etwas von der Bildschirmausgabe abgesetzt, erscheint aber noch in der gleichen Zeile.

Anschließend wird die Hilfsvariable X auf 0 gesetzt und in ihr werden alle Werte des Feldes N() aufsummiert. Bekanntlich ist der Mittelwert einer Stichprobe gleich dem arithmetischen Durchschnitt, d.h. die Summe aller Werte wird durch die Anzahl aller Werte dividiert. Dieses Ergebnis wird der Variablen XQ zugeordnet und anschließend in Zeile 1100 ausgegeben.

Die Varianz einer Stichprobe errechnet sich schon etwas komplizierter. Hier muß die Formel

$$S^2 = \frac{1}{n-1} * \sum_{i=1}^{n} (x_j - \overline{x})^2$$

angewendet werden, wobei $\overline{x}$ der oben errechnete Mittelwert ist. Wir sehen also, daß wir eine weitere Summe bilden müssen, und ziehen dazu die Variable H1 (Hilfsvariable 1) heran, indem wir zunächst die Summe aus der quadrierten Differenz bilden. Die xj sind in unserem Fall in dem Feld N() abgespeichert. Anschließend führen wir noch die Division durch (n-1) durch und erhalten somit in der Variablen SQ die Varianz.

Die restlichen Zeilen dienen nur der Erhaltung der Anzeige am Bildschirm bis eine Taste gedrückt wird.

Kombinatorik.

Die Kombinatorik findet auch im alltäglichen Leben ihren Anwendungsbereich. Zum Beispiel jeder Lottospieler hat mehr oder weniger mit ihr zu tun, da die Errechnung der Anzahl von Gewinnmöglichkeiten den Gesetzen der Kombinatorik unterliegt.

Permutation / Fakultät

Nachdem in den Zeilen 2000 bis 2060 das Untermenü für die Kombinatorik durchlaufen wurde, beginnt in Zeile 2100 das Errechnen der Permutation bzw. der Fakultät. Beide Aussagen stellen das gleiche Ergebnis dar. Die Anzahl der Permutation N stellt die Möglichkeiten dar, N verschiedene Elemente so oft anzuordnen, daß kein Element doppelt auftaucht z.B. kann man mit den neun Ziffern 1,2,3,....,9 9! = 362880 neunstellige Zahlen bilden.

Der Begriff Fakultät (in der Mathematik dargestellt durch ein '!') ist definiert als die fortgesetzte Multiplikation aller Zahlen bis zu dem angegebenen N. Beispiel:

```
1! = 1
2! = 1*2 = 2
3! = 1*2*3 = 6
4! = 1*2*3*4 = 24
usw.
```

Im Programmstück wird zunächst der Basiswert der Fakultät (N) eingegeben. Anschließend wird die Hilfsvariable FA mit '1' vorbesetzt, da sie von Hause aus 0 wäre und die Multiplikation mit 0 aber zu keinem befriedigendem Ergebnis führt. Dann wird in einer Schleife für alle N iterativ jeweils der nächste Wert mit dem bisherigen Ergebnis multipliziert. Dann wird noch das Ergebnis ausgedruckt und zu einem Programmstück gesprungen, was wieder auf das Drücken irgendeiner Taste wartet.

Kombination ohne Wiederholung

Die Anzahl der Kombinationen ohne Wiederholung ist mathematisch definiert als

$$C^{k}_{n} = \binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

Wir sehen, daß auch hier wieder der Begriff Fakultät erscheint und haben das Programmstück aus den Zeilen 2100 bis 2140 nochmal als Unterprogramm in die Zeilen 11000 bis 11040 dupliziert. Bei den Kombinationen ohne Wiederholung kann man die Anordnung berücksichtigen oder nicht, dies wird in den Zeilen 2220 bis 2260 abgefragt. Zunächst errechnet sich in den Zeilen 2270 bis 2430 die Anzahl der Kombination ohne Wiederholung und ohne Berücksichtigung der Anordnung in dem zunächst einige Zwischenrechnungen durchgeführt werden.

Zuerst wird in der Variablen H2 das Ergebnis von N! abgelegt, anschließend in der Variablen H3 das Ergebnis von (N-K)! und in der Variablen H4 das Ergebnis von K!. Dann wird in der Variablen H2 das endgültige Ergebnis in Zeile 2360 errechnet und anschließend ausgegeben.

Die Anzahl der Kombinationen ohne Wiederholung wird auch als Binominalkoiffizient bezeichnet. Bei den Kombinationen ohne Wiederholung aber mit Berücksichtigung der Anordnung entfällt im Nenner lediglich der Term K!, sodaß nach Berechnung von N! und (N-K)! bereits der entsprechende Wert in Zeile 2490 an die Variable H2 zugewiesen werden kann, anschließend erfolgt nur noch die Bildschirmausgabe.

Kombination mit Wiederholung

Die Vorgehensweise bei der Errechnung der Kombination mit Wiederholung mit und ohne Berücksichtigung der Anordnung mag sich der Leser anhand der Formel

$$C^{w\,k}_{n} = \binom{n+k-1}{k} \quad \text{(lies n+k-1 über k)}$$

und des Programmlistings selber verdeutlichen.

Binominalkoeffizient

Wie bereits erwähnt, entspricht der Binominalkoeffizient der Anzahl der Kombinationen ohne Wiederholung. Dieses Programmstück wurde nochmals ab Zeile 3000 bis Zeile 3150 angefügt, um eine Besonderheit der Programmeditierung zu verdeutlichen. Listen Sie die Zeilen 2200 bis 2430 auf dem Bildschirm und ersetzen Sie bei den benötigten Zeilen, die

für die Erfassung der Daten vom Bildschirm und die Berechnung wichtig sind (2220, 2210 und 2270 bis 2360), die Zeilennummern jeweils durch Zeilennummern 3000, 3010 ... . Drücken Sie nach jeder Zeile RETURN. Durch anschließende Eingabe des Befehls LIST 3000-3999 erscheinen diese Zeile wieder auf dem Bildschirm. Dies ist eine einfache Möglichkeit doppelt oder mehrfach benötigte Programmzeilen zu kopieren. Auch von anderen Programmen, wenn sie die benötigten Zeilen auf den Bildschirm holen, dann das Programm laden, in dem diese erscheinen sollen, und dann für jede Zeile RETURN drücken.

Binominalverteilung

Dieses Kopieren von Programmstücken können Sie gleich im folgenden ab Zeile 4000 anwenden, da zur Errechnung der Binominalverteilung auch wieder der Binominalkoeffizient Verwendung findet. Für die Binominalverteilung müssen die Koeffizienten N und P gegeben sein, wobei sich dann automatisch der benötigte Wert Q aus 1-P errechnet. Zusätzlich muß noch der Funktionswert für die Verteilungsfunktion (X) erfaßt werden, dann ergibt sich die Binominalverteilung nach der Formel

$$f(x) = \binom{n}{x} p^{x} q^{n-x}$$

was in der Variablen F gespeichert wird. Der Mittelwert ergibt sich für eine Binominalverteilung nach der Formel N * P und die Varianz aus N * P * Q.

An dieser Stelle wollen wir es mit der Mathematik bewenden lassen. Aber mit ein bischen Mathematik lernt man besser seinen Rechner kennen, da der Computer ursprünglich entwickelt wurde um mathematische Berechnungen schneller und exakter durchführen zu können.

STATISTIK **10 - 11040**

Variablen:

Name	Typ	Bereich	Bedeutung
A	H	0...9	Wert von A$
A$	H	1 Zeichen	Warteschleife
FA	P	Integer	Fakultät
H1	H	Dezimalzahl	HV für Summation bei Varianz
H2	H	Dezimalzahl	HV für Bin.koeff.
H3	H	Dezimalzahl	HV für Bin.koeff.
H4	H	Dezimalzahl	HV für Bin.koeff.
K	K	Integer	Binominalkoeffizient
N	G	Integer	Stichprobengröße
P	G	Dezimalzahl	Binominalverteilung
Q	G	Dezimalzahl	Binominalverteilung
SQ	G	Dezimalzahl	Varianz einer Stich-probe
X	H	Integer	HV für Summation
	G	Dezimalzahl	Funktionswert für Binominalverteilung
XQ	G	Dezimalzahl	Mittelwert einer Stichprobe

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
N	0...255	G	Dezimalzahlen	Stichprobenwerte

Dateien :

#	Name	T	Bemerkung
-	keine		

Unterprogrammaufrufe :

in	nach	Zweck
2280	11000	Fakultät berechnen
2310	11000	Fakultät berechnen
2340	11000	Fakultät berechnen
2450	11000	Fakultät berechnen
2480	11000	Fakultät berechnen
2690	11000	Fakultät berechnen
2720	11000	Fakultät berechnen
2750	11000	Fakultät berechnen
3030	11000	Fakultät berechnen
3060	11000	Fakultät berechnen
3090	11000	Fakultät berechnen
4050	11000	Fakultät berechnen
4080	11000	Fakultät berechnen
4110	11000	Fakultät berechnen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
keine			

2

Der Grafikzeichensatz

2. Der Grafik-Zeichensatz

Wie bisher alle Commodore-Rechner hat auch der Commodore 64 den Grafik-Zeichensatz, mit dem sehr viele (wenn auch nicht alle) grafischen Anwendungen unterstützt werden. Die folgenden Kapitel sind diesen grafischen Sonderzeichen gewidmet. Zunächst sollen die vier grafischen Zeichen Kreuz, Pik, Herz, Karo an zwei Beispielen dargestellt werden. Zum einen wird ein Unterprogramm zur relativen Kartendarstellung am Bildschirm aufgezeigt und anschliessend werden mit diesem Unterprogramm ein Spiel (POKER) erstellt. Da sich die hochauflösende Grafik im Normalfall nicht mit Text zusammen verarbeiten läßt, soll auch ein Programmkomplex für Balkendiagramme ohne hochauflösende Grafik beschrieben werden.

Die grafischen Zeichen gestalten sich bei dem Commodore 64 um einiges interessanter als bei den Commodore der Serien 2000/3000/4000/8000, da sie auch noch in unterschiedlichen Farben dargestellt werden können.

2.1 KARTEN - Ein Unterprogramm zur positionierten Darstellung von Karten am Bildschirm

Übungsziel:

- Nutzung des grafischen Zeichensatzes insbesondere der Kartensymbole
- Aufbau von Unterprogrammen
- Hierarchie von Unterprogrammen
- Testrahmen für Unterprogramme
- Mnemotechnische Zeilennummernvergabe

```
100 POKE53280,0
110 POKE53281,1
120 DATAA,2,3,4,5,6,7,8,9,Z,B,D,K
130 DIMAR$(13)
140 FORI=1TO13
150 READAR$(I)
160 NEXT
170 PRINT"■"
```

```
180 FORJ=1TO4
190 IFJ=1THENF$="■":K$="♣"
200 IFJ=2THENF$="■":K$="♠"
210 IFJ=3THENF$="■":K$="♥"
220 IFJ=4THENF$="■":K$="♦"
230 FORA=1TO13
240 PRINT"■■■■■■■■■■■■■";
250 GOSUB10000
260 GETW$:IFW$=""THEN260
270 NEXTA
280 NEXTJ
290 END
10000 REM*************************************************************
10010 REM*          KARTEN AUSGEBEN                                  *
10020 REM*************************************************************
10030 AR$=AR$(A)
10040 AR$=AR$(A)
10100 REM **********************
10110 REM *****  1.ZEILE *****
10120 REM **********************
10130 PRINT"■┌─────────┐■■■■■■■■■■";
10200 REM **********************
10210 REM *****  2.ZEILE *****
10220 REM **********************
10230 PRINT"│ "F$;AR$;"    ";AR$;"■ │■■■■■■■■■";
10300 REM **********************
10310 REM *****  3.ZEILE *****
10320 REM **********************
10330 IFA=3THENGOSUB15100:GOTO10360
10340 IFA>5THENGOSUB15200:GOTO10360
10350 GOSUB15000
10360 PRINT"│■■■■■■■■■";
10400 REM **********************
10410 REM *****  4.ZEILE *****
10420 REM **********************
10430 IFA=2THENGOSUB15100:GOTO10460
10440 IFA=4ORA=5ORA>7THENGOSUB15200:GOTO10460
10450 GOSUB15000
10460 PRINT"│■■■■■■■■■";
10500 REM **********************
10510 REM *****  5.ZEILE *****
10520 REM **********************
10530 IFA=1ORA=3ORA=5ORA=9THENGOSUB15100:GOTO10570
10540 IFA=7THENGOSUB15300:GOTO10570
10550 IFA=6ORA>9THENGOSUB15200:GOTO10570
10560 GOSUB15000
10570 PRINT"│■■■■■■■■■";
10600 REM **********************
10610 REM *****  6.ZEILE *****
10620 REM **********************
10630 IFA=2THENGOSUB15100:GOTO10660
```

```
10640 IFA=4ORA=5ORA>7THENGOSUB15200:GOTO10660
10650 GOSUB15000
10660 PRINT"█████████|";
10700 REM ********************
10710 REM *****  7.ZEILE *****
10720 REM ********************
10730 IFA=3THENGOSUB15100:GOTO10760
10740 IFA>5THENGOSUB15200:GOTO10760
10750 GOSUB 15000
10760 PRINT"█████████|";
10800 REM ********************
10810 REM *****  8.ZEILE *****
10820 REM ********************
10830 PRINT"| "F$;AR$;"    ";AR$;"█ █████████|";
10900 REM ********************
10910 REM *****  9.ZEILE *****
10920 REM ********************
10930 PRINT"█L_______J";
10940 RETURN
15000 REM**********************************************************
15010 REM* ZEILEN MIT UNTERSCHIEDLICHER ZAHL VON EINTRAEGEN *
15020 REM**********************************************************
15030 REM*****                  KEIN EINTRAG                  *****
15040 PRINT"█|         |";
15050 RETURN
15100 REM*****                  EIN EINTRAG                   *****
15110 PRINT"█| ";F$;"   ";K$;"   ";"█ |";
15120 RETURN
15200 REM*****                  ZWEI EINTRAEGE                *****
15210 PRINT"█| ";F$;" ";K$;" ";K$;" ";"█ |";
15220 RETURN
15300 REM*****                  DREI EINTRAEGE                *****
15310 PRINT"█| ";F$;" ";K$;K$;K$;" ";"█ |";
15320 RETURN
```

Commodore bietet von jeher durch seine vier grafischen Zeichen Kreuz, Pik, Herz und Karo die Möglichkeit, Kartenspiele mit dem Computer möglichst echt zu gestalten. Die Symbole allein nützen einem wenig, wenn man für jedes Programm die Darstellung von Karten neu entwickeln muß. Das vorliegende Unterprogramm erzeugt Karten, mit einer gewünschten Kartenfarbe und Kartenhöhe die von einem vorher angesprungenen Bildschirmpunkt (durch Cursorpositionierung) nach rechts und unten ausgegeben werden. Um eine freie Positionierung innerhalb des Bildschirms zu ermöglichen, sind alle PRINT-Befehle auf den Beginn (linke obere Ecke

der Karte) abgestimmt. Alle Bewegungen des Cursors sind relativ zu diesem Punkt zu sehen.

Zunächst einmal muß man sich überlegen, wie hoch und wie breit eine Karte in Zeichen sein darf. Zunächst die Breite: Für den Rahmen rechts und links muß man zwei Zeichen vorsehen. Da zusätzlich mit den Rahmenzeichen keine weiteren Angaben gemacht werden können, muß man für die Kennzeichnung der Karte (z.B. As, 2, 3, 4) eine weitere Stelle für jede Seite vorsehen und noch drei weitere Stellen insgesamt zum Darstellen der einzelnen Zeichen (Kreuz, Pik, Herz, Karo) auf der Karte. Dies ergibt insgesamt neun Zeichen, sodaß nebeneinander auf dem Bildschirm bis zu fünf Karten dargestellt werden können. Durch die Breitenaufteilung ergibt sich auch schon eine optisch vertretbare Höhe mit neun Zeilen. Der Rahmen muß mitangegeben werden, damit man eine Karte auch vom Hintergrund unterscheiden kann, da wir auch für Kreuz und Pik die Farbe Schwarz und für Herz und Karo die Farbe Rot heranziehen wollen. Würde man für jedes Zeichen eine eigene Hintergrundfarbe wählen, so würde die Programmierung viel zu komplex. Wir stellen daher alle Karten mit einem schwarzen Rand auf weißem Untergrund dar, die Kartenfarbe ist auch Weiß.

Die Karten werden zeilenweise generiert, wobei nach jeder Zeile ein Rücksetzen des Cursors auf die erste Spalte erfolgt, womit die Zeile unter der Letzten ausgegebenen angewählt wird.

Das Unterprogramm teilt sich in zwei Bereiche auf: Die Zeilen 10000 bis 10940 stellen den Hauptteil des Unterprogrammes mit den PRINT-Befehlen. Ab Zeile 15000 bis 15320 befinden sich noch einige Unterprogramme zum unterschiedlichen Eindruck von Kartensymbolen. Wenn man sich eine Karte näher betrachtet, kann man bei zeilenweisem Vorgehen feststellen, daß in einer Zeile entweder keine Einträge von Kreuz, Pik usw. sind, oder ein Eintrag oder zwei Einträge. Für unseren speziellen Fall müssen auch z.B. für die '7' noch drei Einträge in einer Zeile untergebracht werden, da sonst die Kartenhöhe mit neun Zeilen nicht eingehalten werden könnte. Die Unterprogramme ab Zeile 15000 korrespondieren in ihrer Hunderterstelle und drucken eine Kartenzeile mit entsprechender Anzahl von Kreuz, Pik usw. (festgehalten in der Variablen K$). In der Variablen F$ wird jeweils die Farbe der Karte (Schwarz oder Rot) festgehalten. Der Rahmen selbst wird schwarz dargestellt, daher auch jeweils die Farbumschaltung vor einem Randzeichen.

Gehen wir nun die einzelnen Zeilen bei der Ausgabe einer Karte durch. Auch in dem Bereich 10000 bis 10940 sind die

Hunderterstellen der Zeilennummer korrespondierend zu der Zeile der Karte vergeben. Solche mnemotechnischen Tricks gestatten einem das schnelle Auffinden von Programmstellen, ohne erst lange im Listing suchen zu müssen. Die erste Zeile ist zwangsläufig bei allen Karten gleich und beinhaltet den oberen Rand sowie die beiden Ecken. Aber bereits in der zweiten Zeile unterscheiden sich die Ausgaben. Hier werden die Kartenarten ausgegeben. Dazu wird zunächst die Farbe zwischen Schwarz und Rot gegebenenfalls gewechselt, dann wird die Kartenart ausgegeben, die in der Variablen AR$ an das Unterprogramm übergeben wurde. Dies sind die Bezeichnungen A,2,3,4,5,6,7,8,9,Z,B,D,K. Ab der dritten Zeile bis zur siebten Zeile erfolgt die Ausgabe der Kartensymbole. Diese Unterprogrammstücke sind immer gleich aufgebaut: Zunächst wird gefragt ob für bestimmte Kartenhöhen (die in der Variablen A festgehalten sind) Einträge von Kartenzeichen gedruckt werden müssen. In diesem Fall werden die entsprechenden Unterprogramme angesprungen und anschließend ein Sprung auf die Zeile zum Setzen des Cursors auf den nächsten Ausdruck ausgeführt. Wurden keine Karteneinträge ausgegeben, so durchläuft das Programm immer alle IF-Abfragen und landet schließlich bei dem Unterprogrammaufruf 'GOSUB 15000'. Dort wird eine Kartenzeile mit 0 Einträgen gedruckt.

Gehen wir jedoch die Zeilen drei bis sieben der Reihe nach durch. Ist die zu druckende Karte eine '3', so muß in der dritten Zeile ein Zeichen (z.B. Herz) ausgegeben werden. Ist der Kartenwert größer als fünf, so müssen in der dritten Zeile zwei Einträge in den Positionen (Spalten) drei und fünf gedruckt werden. An dieser Stelle sei angemerkt, daß im vorliegenden Unterprogramm die Bildkarten (Bube, Dame, König) wie eine Zehnerkarte dargestellt werden, nur mit anderer Kartenart. Dem Leser bleibt es überlassen, hier mit der Grafik sich eigene Bilder zu entwerfen und diese entsprechend in das Unterprogramm einzubauen.

In der vierten Zeile muß für die Kartenhöhe '2' ein Zeichen ausgegeben werden und für die Kartenhöhen '4', '5', oder '7' bis zu 2 Einträge. In der fünften Zeile für die Werte '1', '3', '5' oder '9' ein Eintrag für die '7' drei Einträge und für die Werte '6' und '9' bis 'K' zwei Einträge. Die sechste und siebte Zeile entsprechen umgekehrt wieder den Zeilen vier und drei, da das Kartenbild symmetrisch ist. Ebenso entsprechen die Zeilen acht und neun den Zeilen zwei und eins, wobei in der neunten Zeile der gespiegelte Rand und die anderen Ecken auszugeben sind.

Da größere Programme im Zusammenhang sehr schwer auszutesten sind, geht man hin und testet einzelne Unterprogramme mit einem Testrahmen. Dies soll hier am Beispiel des Un-

terprogramms zur Kartengenerierung deutlich gemacht werden. Das eigentliche Unterprogramm in den Zeilen 10000 bis 15320 ist alleine nicht lauffähig, kann jedoch in dieser Form in jedes andere Programm eingebaut werden, wobei jedoch beachtet werden muß, daß die Variablen F$, K$, A, und AR$ Eingabeparameter dieses Unterprogramms sind und ihre Verwendung dementsprechend in dem umgebenden Hauptprogramm berücksichtigt werden muß.

Gerade für diese Werte ist in den Zeilen 100 bis 290 ein kurzes Testprogramm eingegeben, indem zunächst die Farben des Bildschirms und des Hintergrundes eingestellt werden (Zeile 100 und 110) und in Zeile 120 die verschiedenen Kartenwerte festgelegt werden, die in den Zeilen 130 bis 160 an das Feld AR$ (13) übergeben werden. In Zeile 170 wird der Bildschirm gelöscht und in den Zeilen 180 bis 280 werden für alle vier Kartenarten die Farbe und die Kartenart vorbesetzt und in einer weiteren Schleife in den Zeilen 230 bis 270 die Karten mit den Werten eins bis dreizehn (1=As, 13=König) ausgegeben. Zur Kontrolle jeder einzelnen Karte wurde in Zeile 260 noch eine Warteschleife auf den Druck irgendeiner Taste eingebaut.

Mit diesem kurzen Testprogramm von zwanzig Zeilen kann man jetzt unabhängig von allen anderen Programmen, wo das Unterprogramm später eingesetzt wird, den Ausdruck jeder einzelnen Karte kontrollieren.

KARTEN **100 - 15320**

Variablen:

Name	Typ	Bereich	Bedeutung
A	E	1...13	Nummer der Kartenart
AR$	E	A,1,2,...,D,K	Kartenart
F$	E	'schwarz' o.'rot'	eigentliche Farbe der Karte
I	H	1...13	Laufvariable K.Höhe
J	H	1...4	Laufvariable K.Art
K$	E	Kreuz,Pik,...	'Kartenfarbe'
W$	H	1 Zeichen (Taste)	zum Einlesen ein. Zeichens von der Tastatur

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
AR$	13	G	A,2,3,...,D,K	Zuordnung Kartenart/Kartennummer

Unterprogrammaufrufe :

in	nach	Zweck
250	10000	Ausgabe einer Karte
10330	15100	Ausgabe einer Kartenzeile mit einem Zeichen
10340	15100	Ausgabe einer Kartenzeile mit zwei Zeichen
10350	15100	Ausgabe einer Kartenzeile mit keinem Zeichen
10430	15100	Ausgabe einer Kartenzeile mit einem Zeichen
10440	15100	Ausgabe einer Kartenzeile mit zwei Zeichen
10450	15100	Ausgabe einer Kartenzeile mit keinem Zeichen
10530	15100	Ausgabe einer Kartenzeile mit einem Zeichen
10540	15100	Ausgabe einer Kartenzeile mit drei Zeichen
10550	15100	Ausgabe einer Kartenzeile mit zwei Zeichen
10560	15100	Ausgabe einer Kartenzeile mit keinem Zeichen
10630	15100	Ausgabe einer Kartenzeile mit einem Zeichen
10640	15100	Ausgabe einer Kartenzeile mit zwei Zeichen
10650	15100	Ausgabe einer Kartenzeile mit keinem Zeichen
10730	15100	Ausgabe einer Kartenzeile mit einem Zeichen
10740	15100	Ausgabe einer Kartenzeile mit zwei Zeichen
10750	15100	Ausgabe einer Kartenzeile mit keinem Zeichen

Verzweigungen nach außen :			
in Ze	nach	Bedingung	Bemerkung
290	END	normaler Abschluß	
10940	RETURN	Ende der Ausgabe einer Karte	bei Verwendung als Unterprogramm

2.2 Poker - ein Ein-Mann-Poker-Spiel

In diesem Pokerspiel wollen wir sogleich das in Kapitel 2.1 vorgestellte Unterprogramm zur Ausgabe von Karten verwenden. Nachgebildet ist dieses Pokerspiel den Spielautomaten, wie sie auch in den Automatenräumen von Spielhallen zu finden sind. Bei diesem Spiel werden fünf Karten am Bildschirm ausgegeben und durch geschicktes Nachkaufen kann man sich eine Kartenkonfigeration zusammenkaufen, durch die ein maximaler Gewinn erzielt wird. Das Spiel führt den Kapitalstand automatisch mit.

Übungsziel

- IF-Abfragen (bedingte Verzweigungen)
- mnemo-technische Zeilennummernvergabe
- Unterprogrammaufrufe und -gestaltung
- Nutzung des grafischen Zeichensatzes

```
1000 REM**************************************************
1010 REM*  VORSPANN UND AUSGABE DER ERSTEN 5 KARTEN     *
1020 REM**************************************************
1030 POKE53280,0
1040 POKE53281,1
1050 KA=100
1060 DIMNE(4)
1070 DIMKA(5,2)
1080 DIMNA(4,2)
1090 DATAA,2,3,4,5,6,7,8,9,Z,B,D,K
1100 DIMAR$(13)
1110 FORI=1TO13
1120 READAR$(I)
1130 NEXT
1140 PRINT"■"
```

```
1150 IFKA<-100THEN5190
1160 KA=KA-5
1170 RF=0
1180 S=0
1190 HS=0
1200 FU=0
1210 FL=0
1220 DR=0
1230 VI=0
1240 P2=0
1250 P1=0
1260 FORI=1TO5
1270 ON I GOSUB 20100,20200,20300,20400,20500
1280 KA(I,1)=INT(RND(5)*13)+1
1290 KA(I,2)=INT(RND(3)*4)+1
1300 FORJ=I-1TO1STEP-1
1310 IFKA(I,1)=KA(J,1)ANDKA(I,2)=KA(J,2)THEN1280
1320 NEXTJ
1330 GOSUB10000
1340 NEXTI
2000 REM*****************************************************
2010 REM*          NACHKAUFEN                                 *
2020 REM*****************************************************
2030 PRINT"■■    1        2        3        4        5"
2040 PRINT"■DEIN KAPITAL";KA
2050 PRINT"■MOECHTEST DU NEUE KARTEN (J/N)"
2060 GETA$:IFA$=""THEN2060
2070 IFA$="J"THEN2100
2080 IFA$="N"THEN3000
2090 GOTO2060
2100 PRINT"■■WIEVIELE KARTEN ? ";
2110 GETA$:IFA$=""THEN2110
2120 PRINTA$
2130 N=VAL(A$)
2140 KA=KA-N
2150 IFN<1ORN>4THENPRINT"■MEHR ALS 4 GEHT NICHT":GOTO2110
2160 FORK=1TON
2170 NA(K,1)=INT(RND(5)*13)+1
2180 NA(K,2)=INT(RND(3)*4)+1
2190 FORM=K-1TO1STEP-1
2200 IFNA(K,1)=NA(M,1)ANDNA(K,2)=NA(M,2)THEN2170
2210 NEXTM
2220 FORM=1TO5
2230 IFNA(K,1)=KA(M,1)ANDNA(K,2)=KA(M,2)THEN2170
2240 IFNA(K,1)=KA(M,1)ANDNA(K,2)=KA(M,2)THEN2170
2250 NEXTM
2260 NEXTK
2270 PRINT"■BITTE DRUECKE JETZT DIE ENT-
2280 PRINT"SPRECHENDEN ZIFFERNTASTEN ZU DEN
2290 PRINT"NUMMERN UNTER DEN KARTEN
2300 FORK=1TON
```

```
2310 GETA$:IFA$=""THEN2310
2320 B=VAL(A$)
2330 IFB<1ORB>5THEN2320
2340 NE(K)=B
2350 NEXTK
2360 FORL=1TON
2370 KA(NE(L),1)=NA(L,1)
2380 KA(NE(L),2)=NA(L,2)
2390 ON NE(L) GOSUB 20100,20200,20300,20400,20500
2400 I=NE(L)
2410 GOSUB10000
2420 NEXTL
2430 PRINT
3000 REM*****************************************************
3010 REM*                    AUSWERTUNG                     *
3020 REM*****************************************************
3030 REM *****    FLASH    *****
3040 FORI=1TO5
3050 FORJ=ITO5
3060 IFKA(I,2)<>KA(J,2)THEN3100:REM KEIN FLASH
3070 NEXTJ
3080 NEXTI
3090 FL=1
3100 REM *****   STRASSE  *****
3110 REM *****   SORTIEREN  *****
3120 FORI=1TO5
3130 MI=14
3140 FORJ=ITO5
3150 IFKA(J,1)<MITHENMI=KA(J,1):WO=J
3160 NEXTJ
3170 H=KA(WO,1)
3180 KA(WO,1)=KA(I,1)
3190 KA(I,1)=H
3200 NEXTI
3210 REM *****   STRASSE  JA/NEIN  *****
3220 IFKA(1,1)+1<>KA(2,1)THEN3260
3230 IFKA(2,1)+1<>KA(3,1)THEN3260
3240 IFKA(3,1)+1<>KA(4,1)THEN3260
3250 IFKA(1,1)<>1THENS=1
3260 REM *****   HOECHSTE STRASSE JA/NEIN  *****
3270 IFKA(1,1)=1ANDKA(2,1)=10ANDKA(5,1)=13THENHS=1
3280 REM *****   ROYAL FLASH  JA/NEIN  *****
3290 IFHSANDFLTHENRF=1:GOTO4000
3300 IFHSORSORFLTHEN4000
3310 REM *****   VIERER JA/NEIN  *****
3320 IFKA(1,1)=KA(4,1)ORKA(2,1)=KA(5,1)THENVI=1:GOTO4000
3330 REM *****   DREIER JA/NEIN  *****
3340 IFKA(1,1)=KA(3,1)THENDR=1
3350 IFKA(2,1)=KA(4,1)THENDR=2
3360 IFKA(3,1)=KA(5,1)THENDR=3
3370 REM *****   FULL HOUSE JA/NEIN  *****
```

```
3380 IFDR=1ANDKA(4,1)=KA(5,1)THENFU=1
3390 IFDR=3ANDKA(1,1)=KA(2,1)THENFU=1
3400 IFDRORFUTHEN4000
3410 REM *****   2 PAARE JA/NEIN  *****
3420 IFKA(1,1)=KA(2,1)ANDKA(3,1)=KA(4,1)THENP2=1
3430 IFKA(1,1)=KA(2,1)ANDKA(4,1)=KA(5,1)THENP2=1
3440 IFKA(2,1)=KA(3,1)ANDKA(4,1)=KA(5,1)THENP2=1
3450 IFP2THEN4000
3460 REM *****   1 PAAR JA/NEIN  *****
3470 IFKA(1,1)=KA(2,1)THENP1=1
3480 IFKA(2,1)=KA(3,1)THENP1=1
3490 IFKA(3,1)=KA(4,1)THENP1=1
3500 IFKA(4,1)=KA(5,1)THENP1=1
4000 REM*****************************************************
4010 REM*          GEWINN FESTSTELLEN                        *
4020 REM*****************************************************
4030 PRINT"■■■■■■■■■■■■■■■■■■DU HAST ";
4040 IFRFTHENPRINT"EINEN■ROYAL FLASH":GE=1000:GOTO4140
4050 IFFLANDSTHENPRINT"EINEN STRAIGHT FLASH":GE=750:GOTO
4060 IFVITHENPRINT"EINEN■VIERER":GE=500:GOTO4140    4140
4070 IFFLTHENPRINT"EINEN FLASH":GE=350:GOTO4140
4080 IFFUTHENPRINT"EIN FULL HOUSE":GE=250:GOTO4140
4090 IFSTHENPRINT"EINEN STRAIGHT":GE=100:GOTO4140
4100 IFDRTHENPRINT"EINEN DREIER":GE=50:GOTO4140
4110 IFP2THENPRINT"ZWEI PAARE":GE=25:GOTO4140
4120 IFP1THENPRINT"EIN PAAR":GE=5:GOTO4140
4130 PRINT"NICHTS":GE=0
4140 PRINT"DEIN KAPITAL VERMEHRT SICH UM "GE"PUNKTE";
4150 KA=KA+GE
4160 PRINT"                                                ";
4170 PRINT"                                                "
4180 PRINT"DEIN KAPITAL BETRAEGT JETZT";KA
5000 REM*****************************************************
5010 REM*    SPIELENDE / ABSCHLUSS                           *
5020 REM*****************************************************
5030 PRINT"WEITERES SPIEL ?"
5040 GETA$:IFA$=""THEN5040
5050 IFA$="J"THEN1140
5060 IFA$="N"THEN5080
5070 GOTO5040
5080 POKE53280,7
5090 POKE53281,7
5100 PRINT"■■■■DU HAST JETZT EIN KAPITAL VON ";KA
5110 IFKA<0THENPRINT"■■■■■DRAUFGEZAHLT ? WIE ?":END
5120 IFKA<50THENPRINT"■■■■■NICHT BESONDERS":END
5130 IFKA<150THENPRINT"■■■■■A BISSL WAS ISTS A":END
5140 IFKA<200THENPRINT"■■■■■BEACHTLICH":END
5150 IFKA<500THENPRINT"■■■■■HERVORRAGEND":END
5160 IFKA<1000THENPRINT"■■■■■ICH WUESSTE DA EINE GUTE
5170 IFKA<1000THENPRINT"EINNAHMEQUELLE":END ZUSAETZLICHE
5180 IFKA>1000THENPRINT"■■■■■■■■■UND DA SPIELST DU NUR
                                        MIT MIR ???!!!???":END
```

```
5190 POKE53280,7
5200 POKE53281,7
5210 PRINT"■■■DU HAST JETZT EIN KAPITAL VON ";KA  !!!!!":END
5220 PRINT"■■■■■ICH KANN DIR LEIDER KEINEN KREDIT MEHR  GEBEN
10000 REM*************************************************************
10010 REM*          KARTEN AUSGEBEN                                    *
10020 REM*************************************************************
10030 A=KA(I,1)
10040 AR$=AR$(A)
10050 IFKA(I,2)=1THENF$="■":K$="♣"
10060 IFKA(I,2)=2THENF$="■":K$="♠"
10070 IFKA(I,2)=3THENF$="■":K$="♥"
10080 IFKA(I,2)=4THENF$="■":K$="♦"
10100 REM **********************
10110 REM *****  1.ZEILE *****
10120 REM **********************
10130 PRINT"■┌──────┐■■■■■■■■";
10200 REM **********************
10210 REM *****  2.ZEILE *****
10220 REM **********************
10230 PRINT"│ "F$;AR$;"    ";AR$;"■ ■■■■■■■■";
10300 REM **********************
10310 REM *****  3.ZEILE *****
10320 REM **********************
10330 IFA=3THENGOSUB15100:GOTO10360
10340 IFA>5THENGOSUB15200:GOTO10360
10350 GOSUB15000
10360 PRINT"■■■■■■■■";
10400 REM **********************
10410 REM *****  4.ZEILE *****
10420 REM **********************
10430 IFA=2THENGOSUB15100:GOTO10460
10440 IFA=4ORA=5ORA>7THENGOSUB15200:GOTO10460
10450 GOSUB15000
10460 PRINT"■■■■■■■■";
10500 REM **********************
10510 REM *****  5.ZEILE *****
10520 REM **********************
10530 IFA=1ORA=3ORA=5ORA=9THENGOSUB15100:GOTO10570
10540 IFA=7THENGOSUB15300:GOTO10570
10550 IFA=6ORA>9THENGOSUB15200:GOTO10570
10560 GOSUB15000
10570 PRINT"■■■■■■■■";
10600 REM **********************
10610 REM *****  6.ZEILE *****
10620 REM **********************
10630 IFA=2THENGOSUB15100:GOTO10660
10640 IFA=4ORA=5ORA>7THENGOSUB15200:GOTO10660
10650 GOSUB15000
10660 PRINT"■■■■■■■■";
10700 REM **********************
```

```
10710 REM *****  7.ZEILE *****
10720 REM ********************
10730 IFA=3THENGOSUB15100:GOTO10760
10740 IFA>5THENGOSUB15200:GOTO10760
10750 GOSUB 15000
10760 PRINT"■■■■■■■■■|";
10800 REM ********************
10810 REM *****. 8.ZEILE *****
10820 REM ********************
10830 PRINT"| "F$;AR$;"    ";AR$;"■ ■■■■■■■■■|";
10900 REM ********************
10910 REM *****  9.ZEILE *****
10920 REM ********************
10930 PRINT"■L________|";
10940 RETURN
15000 REM*******************************************************
15010 REM* ZEILEN MIT UNTERSCHIEDLICHER ZAHL VON EINTRAEGEN *
15020 REM*******************************************************
15030 REM*****                 KEIN EINTRAG                *****
15040 PRINT"■|         |";
15050 RETURN
15100 REM*****                 EIN EINTRAG                 *****
15110 PRINT"■| ";F$;"   ";K$;"   ";"■ |";
15120 RETURN
15200 REM*****                 ZWEI EINTRAEGE              *****
15210 PRINT"■| ";F$;"  ";K$;"  ";K$;"  ";"■ |";
15220 RETURN
15300 REM*****                 DREI EINTRAEGE              *****
15310 PRINT"■| ";F$;"  ";K$;K$;K$;"  ";"■ |";
15320 RETURN
19999 END
20000 REM*******************************************************
20010 REM*             KARTE POSITIONIEREN                     *
20020 REM*******************************************************
20100 PRINT"■■";
20110 RETURN
20200 PRINT"■■■■■■■■■";
20210 RETURN
20300 PRINT"■■■■■■■■■■■■■■■■■■";
20310 RETURN
20400 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■";
20410 RETURN
20500 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■";
20510 RETURN
60000 SAVE"@:POKER",8
60010 OPEN1,8,15
60020 INPUT#1,A,B$,C,D
60030 PRINTA;B$;C;D
60040 CLOSE1
60050 END
```

Zeilenbereiche

1000 - 1340	Programmvorspann und Ausgabe der ersten fünf Karten
2000 - 2430	Nachkaufen von Karten
3000 - 3500	Auswertung der gezogenen Karten
4000 - 4180	Gewinn feststellen und Ausgabe der Gewinnart, der Gewinnsumme und des neuen Kapitals
5000 - 5220	Spielende und Abschluß
10000 - 10940	Unterprogramm zur Darstellung der Karten
15000 - 15320	Unterprogramme, die zur Darstellung der Karten benötigt werden, (vergleiche Kapitel 2.1)
20000 - 20510	Unterprogramme zum Positionieren der fünf Karten
60000 - 60050	Hilfsprogramm zum Abspeichern der neuen Programmversion

Vorspann und Ausgabe der ersten fünf Karten

Zunächst werden auch in diesem Programm wieder die Farben für Bildschirm und Rand gesetzt (Zeilen 1030 und 1040) dann wird das zur Verfügung stehende Kapital mit 100 Einheiten vorbesetzt und die benötigten Felder werden dimensioniert. (vergl. Variablenliste). Im Weiteren werden die Kartenwerte (A,2,...,D,K) vorgegeben, sowie ein Feld zum Speichern dieser Werte dimensioniert und besetzt (Zeilen 1090 bis 1130). Dann wird der Bildschirm gelöscht und abgefragt ob mehr als 100 Kapitaleinheiten Schulden beim Spieler bestehen, in diesem Falle wird eine Meldung ausgegeben (auf blauem Bildschirm), daß dem Spieler kein Kredit mehr gewährt wird und das Spiel wird abgebrochen (Zeilen 5190 bis 5220).

Die Zeile 1140 ist auch gleichzeitig Ansprungstelle für jedes weitere Spiel. Im weiteren Verlauf wird in Zeile 1160 das Kapital um fünf Einheiten verringert (Spieleinsatz pro Spiel) und diverse Variablen, die die späteren Gewinnmöglichkeiten festhalten sollen, mit '0' vorbesetzt. In der Schleife in den Zeilen 1260 bis 1340 werden die ersten fünf Karten am Bildschirm ausgegeben, dazu wird zunächst aufgrund der Laufvariablen I in die Unterprogramme ab 20000 gesprungen, wobei die Hunderter-Stelle der Zeilenzahl jeweils mit der Nummer der Karte übereinstimmt. Diese Unterprogramme positionieren den Cursor entsprechend für die auszugebende Karte. In den Zeilen 1280 und 1290 werden Kartenhöhe und Kartenart bestimmt, und in den Zeilen 1300 bis 1320 wird festgestellt ob diese Karte bereits

gezogen wurde.

Dies ist natürlich kein korrektes Mischen von Karten, aber ein korrektes Mischprogramm zu erstellen bleibt dem Leser überlassen. Vielleicht versuchen Sie es mit einer Zeichenreihe von 104 Stellen wo jeweils zwei benachbarte Stellen Kartenhöhe und Kartenart anzeigen. Diese Zeichenreihe verarbeiten Sie dann mit RIGHT$(,), LEFT$(,) und MID$(,,) wie es auch dem natürlichen Mischen entspricht. Setzen Sie für die Länge der abzuschneidenden Zeichenreihe jeweils eine Zufallszahl - die natürlich gerade sein muß - zwischen 0 und 104 ein, vielleicht mit dem Befehl :

```
ZU = (INT( RND(5) *52 ) + 1) *2.
```

Wenn Sie die Karten jedoch korrekt mischen werden Sie feststellen, daß das Spiel sehr schnell uninteressant wird, da Gewinnkombinationen viel seltener auftauchen, als bei der programmierten Zufallsauswahl.

Die Zeilen 1300 bis 1320 sind unbedingt notwendig, wenn der Computer nicht zum Falschspieler werden soll. Zeile 1130 veranlaßt die Ausgabe der 'gezogenen' Karte.

Nachkaufen von Karten

Die Zeilen 2000 bis 2430 beschäftigen sich mit dem Nachkaufen von Karten. Die Karten werden in Zeile 2030 durchnumeriert, und im folgenden wird das Kapital angezeigt und gefragt, ob neue Karten gezogen werden sollen oder nicht. Die Zeilen 2060 bis 2090 beinhalten eine Plausibilitätsprüfung mit entsprechender Verzweigung aufgrund oben angeführter Abfrage.

Als Nächstes wird gefragt, wieviele Karten nachgekauft werden sollen (falls oben ein 'j' eingegeben wurde). Ansonsten wird das Programm bei der Gewinnermittlung fortgesetzt. In Zeile 2150 wird geprüft ob zwischen '1' und '4' Karten nachgezogen werden sollen. Wie üblich ist die Ziehung eines kompletten neuen Kartensatzes nicht möglich. In Zeile 2140 wird noch das Kapital um die Einheiten verringert, wie auch die Anzahl der gezogenen Karten beträgt. Soll der Computer keine Strafe für Falscheingaben berechnen, so sind die Zeilen 2140 und 2150 zu vertauschen.

In den Zeilen ab 2160 werden entsprechend Karten nachgezogen. Die gezogenen Karten werden natürlich mit den bereits am Bildschirm angezeigten und auch eventuell bereits nach-

gezogenen Karten verglichen, ob der Computer nicht 'schummelt'. In den Zeilen ab 2270 wird abgefragt, welche Karten der zuerst gezogenen ersetzt werden sollen. Dies geschieht durch einfaches Drücken der entsprechenden Zifferntasten 1 - 5 (ohne RETURN, da eine GET-Abfrage eingebaut ist). Die Zeile 2330 enthält eine Plausibilitätsprüfung, daß keine Ziffern größer als fünf eingegeben werden dürfen. Im weiteren werden die entsprechenden Karten ausgetauscht (Zeilen 2370 und 2380) und ausgegeben (Zeile 2390 bis 2410).

Auswertung der gezogenen Karten (mit **Nachkaufen**)

- Flush

In den Zeilen 3040 bis 3090 wird abgeprüft ob ein sogenannter Flush vorliegt, der bekanntermaßen aus fünf Karten in der gleichen Farbe besteht. Da die Kartenfarbe intern im System durch Zahlen dargestellt wird (2.Element der Matrix KA(,), muß festgestellt werden, ob diese Elemente bei allen Karten gleich sind. Aus Rechenzeitgründen wird sofort auf das Prüfen der Straße gesprungen sobald zwei paarweise verschiedene Karten auftreten.

- Straße

Aus unsortierten Karten festzustellen ob eine Straße vorliegt, ist ein schwieriges Unterfangen. Einfacher wird es, wenn man die Kartenhöhen in Ihrer Reihenfolge entsprechend sortiert, dazu wird ein einfaches Sortierverfahren (MINSORT) herangezogen, was die Kartenhöhen in aufsteigender Reihenfolge sortiert (1.Element der Variablen KA(,)). Mehr über Sortierprogramme finden Sie in 'Basic ohne Probleme' Band 3 Kapitel 1.4.5.

Nachdem die Karten in aufsteigender Reihenfolge sortiert sind, muß abgeprüft werden, ob eine Straße vorliegt. Dies ist nicht der Fall, wenn der Wert der ersten Karte um eins erhöht verschieden vom Wert der zweiten Karte ist bzw. die gleiche Aussage bei den Kartengruppierungen 2./3., 3./4. und 4./5. Karte auftritt. Da es aber auch eine Straße mit 10, Bube, Dame , König und As gibt, das As aber den Wert '1' hat, muß noch gesondert abgeprüft werden, ob die erste Karte ungleich eins ist.

Die höchste Straße (10 bis As) liegt also vor, wenn in der sortierten Matrix KA(,) das erste Element eine '1' enthält und die Elemente zwei bis fünf die Werte '10' bis '13'. In diesem Fall wird der Merker für die höchste Straße (HS)

auf '1' gesetzt.

- Royal-Flush

Ein Royal-Flush liegt dann, und nur dann vor, wenn die höchste Straße vorliegt und zugleich ein Flush (10 bis As in der gleichen Farbe).

Liegt jetzt bereits eine Gewinnauswertung (Straße, Flush oder Royal-Flush) vor, so wird die weitere Auswertung übersprungen und gleich zur Feststellung der Gewinnsumme übergegangen.

- Vierer (Vierling,Four of a Kind)

Da in diesem Stadium auf jeden Fall die Kartenhöhen sortiert sind, braucht für einen Vierling nur noch festgestellt zu werden, ob die Kartenhöhe der ersten Karte gleich der vierten Karte bzw. der zweiten Karte gleich der fünften Karte ist. In diesem Falle wird die Variable VI als Flag (Merker) mit '1' besetzt.

- Drilling (Dreier,Three of a Kind)

Etwas komplexer als beim Vierer liegt die Sache beim Drilling. Hier müssen die Kartengruppen 1/3 bzw. 2/4 bzw. 3/5 gleich sein. Durch die sortierte Vorgabe ist damit gleich eingeschlossen, daß die Karten 2 bzw. 3 bzw. 4 auch die gleiche Höhe aufweisen. Etwas vorausschauend gedacht wird in der Variablen DR nicht nur abgespeichert ob ein Drilling vorhanden ist oder nicht, sondern auch die Lage innerhalb dieser sortierten Reihenfolge. Dies ermöglicht uns eine einfacherere Feststellung beim

- Full House

Naturgemäß kann ein Full House (Drilling plus Zwilling) nicht mehr dann vorliegen, wenn innerhalb der sortierten Reihenfolgen die Karten zwei, drei und vier gleich sind. Prinzipiell ist dann kein weiterer Zwilling mehr möglich. Deshalb wird die Variable DR nur noch abgeprüft, wenn eine '1' oder '3' vorliegt. In diesem Falle müssen entweder die Kartenhöhen 4/5 (für DR=1) oder Kartenhöhen 1/2 (für DR=3) übereinstimmen.

Liegt ein Drilling oder ein Full House vor, so wird die Auswertung ob ein bzw. zwei Paare vorliegen übersprungen. Ein Sprung zur Gewinnauswertung sofort nach dem Drilling ist nicht ratsam, da aus einem Drilling ja noch immer ein Full House werden kann.

- Zwei Paare

Auch hier ist die sortierte Reihenfolge der Kartenhöhen wieder von Vorteil, es müssen jeweils die möglichen Kartengruppen 1/2 mit 3/4 bzw. 1/2 mit 4/5 bzw. 2/3 mit 4/5 überoprüft werden. Liegen zwei Paare vor, so kann man auch die Prüfung auf ein Paar überspringen.

- Ein Paar (Zwilling)

Für ein Paar müssen lediglich die Kartenhöhe der Paare 1/2, 2/3, 3/4 und 4/5 auf Gleichheit geprüft werden.

Gewinn feststellen

In Abhängigkeit von den in den Zeilen 3000 bis 3500 gesetzten Flags wird nun eine Bildschirmmeldung nach der Art des Gewinns in den Zeilen 4030 bis 4130 ausgegeben. Eine Änderungsmöglichkeit für den Leser besteht darin, eine einzige Variable, nennen wir sie G, mit den Werten '1' bis '10' entsprechend der Gewinnhöhe vorzubesetzen und dann mit einem 'ON G GOTO'-Befehl eine Verzweigung durchzuführen, die nicht so viel Rechenzeit beansprucht, wie die derzeit programmierte mit den verschiedenen IF-Abfragen. Man kann sich auch überlegen, wenn die IF-Abfragen in den Zeilen 4040 bis 4120 in umgekehrter Reihenfolge programmiert werden, ob dies eine Rechenzeitersparnis bringt.

In der Variablen GE wird nach jeder Ausgabe der Gewinn festgelegt. Wer sich eine Spielbeschreibung für andere Anwender erstellen will, kann hieraus auch die Gewinntabelle ablesen.

Anschließend wird noch eine Meldung ausgegeben, daß sich das Kapital um 'GE' Punkte vermehrt hat, die Kapitalsummierung durchgeführt, und der neue Kapitalstand ausgegeben.

Spielende / Abschluß

Ab Zeile 5030 fragt der Rechner ob ein weiteres Spiel durchgeführt werden soll. Wenn ja, wird wieder auf Zeile 1140 gesprungen, wenn nein, wird in Abhängigkeit vom erreichten Gewinn, eine kurze Meldung des Rechners ausgegeben.

Es wird immer wieder festgestellt, daß gerade diese - programmtechnisch doch so einfach zu erstellenden - Meldungen die größte Resonanz beim Publikum finden. Rechner mit

künstlicher Intelligenz? Nein - einfachste Programmierung!

Der Ablauf der Zeilen 10000 bis 15320 ist in Kapitel 2.1 beschrieben. Das Karten positionieren ab Zeile 20000 ist auch einfach, da jeweils nur eine Zeichenreihe aus 'Cursor nach rechts' ausgedruckt wird, der noch ein 'HOME' vorangestellt ist. Um die Karten nicht zu sehr an den oberen Bildschirmrand zu bringen, wurde auch noch ein 'Cursor nach unten' eingefügt. Obwohl die Karten nur sieben Zeichen breit sind wird der Cursor jeweils um acht weitere Stellen nach rechts geschoben, um die Karten nicht zu dicht nebeneinander liegen zu lassen und auch den ganzen Bildschirm bis zum rechten Rand auszufüllen.

Wenn Sie immer gegen Ihre Mitspieler gewinnen wollen (Kapitalmäßig) so sorgen Sie beim Nachkauf dafür, daß Sie möglichst einen Flush bzw. Royal-Flush bekommen. Die Wahrscheinlichkeit hierfür ist am größten, wegen der oben beschriebenen Kartenauswahl, die ja kein eigentliches Mischen ist. Der geneigte Leser kann sich zusätzlich zu dem oben beschriebenen Vorschlag zum Mischen von Karten noch überlegen, wie dieser Fehler behoben werden kann. Vielleicht durch ändern der Multiplikatoren '13' und '4' in den Zeilen 1280 und 1290 in entsprechend vorbesetzte Variablen, die vielleicht auch nicht ganzzeilig sein müssen.

Weitere Ergänzung

Spielern ist es sicherlich schon aufgefallen: der Straight Flush fehlt. Bauen sie ihn analog zu den anderen Gewinnen ein! Wann liegt bei den vorgegebenen Daten ein Straight Flush vor? Welche Auswertungen können übersprungen werden, wie ist der Gewinn einzuordnen?

POKER 1000 - 20510

Variablen:

Name	Typ	Bereich	Bedeutung
A	P	1...13	Kartenhöhe für Ausgabe
A$	H	1 Zeichen	zum Einlesen von der Tastatur
B	H	1...5	aktuelle Nummer einer

			neuen Karte
DR	G	0,1,2,3	Merker für Drilling
F$	P	Schwarz,Rot..	Kartenfarbe f. Ausgabe
FL	G	0,1	Merker für Flush
FU	G	0,1	Merker für Full House
H	H	1...13	HV für Sortieren
HS	G	0,1	Merker für höchste Strasse
I	H	Integer	Kartennummer
J	H	Integer	Laufvariable
K	H	Integer	Laufvariable
K$	P	Kreuz,Pik,...	Kartenart für Ausgabe
KA	G	Integer	Spielkapital
M	H	Integer	Laufvariable
MI	H	Integer	HV für Sortieren
N	G	0...4	Anzahl neuer Karten
P1	G	0,1	Merker für ein Paar
P2	G	0,1	Merker für zwei Paare
RF	G	0,1	Merker für Royal Flush
S	G	0,1	Merker für Strasse
VI	G	0,1	Merker für Vierling
WO	H	1...5	HV für Sortieren

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
AR$	1...13	G	A,1,...,D,K	Kartenhöhe
NE	0...4	G	1...5	Nummer der neuen Karten
KA	1...5/ 1...2	G	1...13,1...4	gezogene Karten
NA	1...5/ 1...2	G	1...13,1...4	nachgezogene Karten

Unterprogrammaufrufe :

in	nach	Zweck
1270	20100	Positionieren für 1. Karte
1270	20200	Positionieren für 2. Karte
1270	20300	Positionieren für 3. Karte
1270	20400	Positionieren für 4. Karte
1270	20500	Positionieren für 5. Karte
1330	10000	Karte ausgeben

2390	20100	Positionieren für 1. Karte
2390	20200	Positionieren für 2. Karte
2390	20300	Positionieren für 3. Karte
2390	20400	Positionieren für 4. Karte
2390	20500	Positionieren für 5. Karte
2410	10000	Karte ausgeben
----	-----	----------
		ab Zeile 10000 vergleiche Kapitel 2.1

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
keine			

2.3 Balkendiagramme ohne hochauflösende Grafik

Das Ergebnis irgendwelcher Berechnungen sind sehr häufig Zahlenkolonnen, die für sich selbst natürlich nicht anschaulich sind. Aus diesem Grunde greift man immer wieder auf Diagramme zurück. Dies können Euler-Diagramme, Tortengrafik, Kurven oder auch Balkendiagramme sein. Auf dem Commodore 64 lassen sich Balkendiagramme mit ausreichender Auflösung auch ohne die hochauflösende Grafik darstellen. Dies hat den Vorteil, daß sehr einfach Text und Grafik gemischt werden können, da die hochauflösende Grafik von Hause aus keine Ausgabe von Text gestattet.

Wie auch bei den anderen Commodore-Geräten der Serie 2000 / 3000 / 4000 und 8000 hat auch der Commodore 64 in seinem Grafikzeichensatz die Zeichen, die für eine hohe Auflösung bei Balkendiagrammen benötigt werden. Im Prinzip läßt sich ein Balkendiagramm aus lauter Reverse-Leerzeichen darstellen, bei dem Commodore 64 erhält man auf diese Weise, bei voller Ausnutzung des Bildschirms für die Diagramme, eine Auflösung von horizontal 40 Zeichen und vertikal 24 bzw. 25 Zeichen. Dies ist für eine genaue Darstellung natürlich nicht ausreichend.

Ein Zeichen am Commodore 64 besteht aus 64 einzelnen Punkten die in einer 8 x 8 Matrix angeordnet sind. Durch die Grafikzeichen ist man nun in der Lage von diesen 8 x 8 Punkten jeweils die untersten acht, die untersten sechzehn usw. anzusprechen, sodaß man im Prinzip das Zeichen zei-

lenweise zerlegen kann. Selbiges gilt auch für die linke Spalte, die zwei linken Spalten usw. in der Zeichenmatrix. Dadurch läßt sich für den Commodore 64 eine Auflösung bei Balkengrafiken von maximal 200 bei vertikaler Darstellung und 320 bei horizontaler Darstellung erreichen, wie es auch die hochauflösende Grafik könnte.

In den folgenden vier Unterkapiteln wollen wir die Möglichkeiten für ein vertikales und horizontales Balkendiagramm aufzeigen, sowie ein Programm zur Relativierung der Werte in den Balkendiagrammen und ein Programm zum frei definierbaren Einzeichnen von Balken im Bildschirm.

2.3.1 Vertikale Balkendiagramme

Übungsziel:

- grafische Zeichen benutzen
- IF-Abfragen
- Farbgebung
- Balkendiagramme

Listing siehe nächste Seite

In diesem Kapitel wird ein Beispielprogramm für vertikale Balkendiagramme vorgestellt, das sich selbst immer wieder aufruft und die Balkenhöhe jeweils per Zufallszahl bestimmt. Die Farben werden der Reihe nach vom Computer immer in der gleichen Reihenfolge (8 verschiedene Farben) vergeben. Zunächst setzen wir die Bildschirmhintergrundfarbe und die Farbe für den Rand auf schwarz, und die Farbe der Schrift auf gelb. Da bis zu 40 Balken nebeneinander gedruckt werden können, dimensionieren wir ein Feld A(40) zur Aufnahme der Balkenhöhe. Im weiteren werden zwei Zeichenreihen mit Cursorsteuerfunktionen besetzt: CR$ für Cursor nach rechts mit der Länge 40 und CD$ für Cursor nach unten (Cursor Down) mit 25 Zeichen Länge. Dann wird der Bildschirm gelöscht und zwei Hilfsvariablen (S für Spalte / Z für Zeile), die das unterste Zeichen des jeweiligen Balkens am Bildschirm angeben werden, bestimmt.

Die Positionierung des Cursors zum Zeichnen des jeweiligen Balkens erfolgt in dem Unterprogramm ab Zeile 1000, wobei dort zunächst der Cursor in die 'HOME'-Position gebracht wird (oben links am Bildschirm) und anschließend für die Anzahl der Spalten nach rechts und für die Anzahl der Zeilen nach unten bewegt wird. In dem Beispielprogramm beginnen die Balken jeweils in Zeile 19, da die beiden darunterliegenden Zeilen noch für eine Indizierung vorgesehen

```
100 POKE53280,0
110 POKE53281,0
120 PRINT"■
130 DIMA(40)
140 CR$="■"
150 CD$="■"
160 FORI=1TO6
170 CR$=CR$+CR$
180 CD$=CD$+CD$
190 NEXT
200 CR$=LEFT$(CR$,40)
210 CD$=LEFT$(CD$,25)
220 PRINT"■"
230 S=0
240 Z=22
250 GOSUB1000
260 PRINT"0000000001111111111222222222333333333334";
270 PRINT"1234567890123456789012345678901234567890"
280 FORI=1TO40
290 A(I)=INT(RND(8)*160)
300 REM A(I)=I
310 NEXT
320 FORI=1TO40
330 F=F+1
340 IFF=1THENPRINT"■
350 IFF=2THENPRINT"■
360 IFF=3THENPRINT"■
370 IFF=4THENPRINT"■
380 IFF=5THENPRINT"■
390 IFF=6THENPRINT"■"
400 IFF=7THENPRINT"■"
410 IFF=8THENPRINT"■":F=0
420 S=I-1
430 Z=19
440 GOSUB1000
450 IFA(I)>=8THENPRINT"■ ■";:A(I)=A(I)-8:GOTO450
460 IFA(I)=7THENPRINT"■■";
470 IFA(I)=6THENPRINT"■■";
480 IFA(I)=5THENPRINT"■■";
490 IFA(I)=4THENPRINT"■■";
500 IFA(I)=3THENPRINT"■■";
510 IFA(I)=2THENPRINT"■■";
520 IFA(I)=1THENPRINT"■■";
530 NEXT
540 FORI=1TO10000
550 NEXT
560 RUN
1000 PRINT"■";LEFT$(CR$,S);LEFT$(CD$,Z);
1010 RETURN
60000 SAVE"@:BALKENV",8
```

sind. In der Schleife von Zeile 280 bis Zeile 310 werden die Zufallszahlen für die Balkenhöhe generiert. Setzen Sie Zeile 290 auf Kommentar mit einem REM, und löschen Sie das REM in Zeile 300, so erhalten Sie die grafische Darstellung einer Treppe. In den Zeilen 320 bis 530 befindet sich die FOR...Next-Schleife zur Ausgabe der Balken. Zunächst wird jeweils der Farbparameter (F) um eins erhöht und die entsprechende Farbe durch einen Steuerbefehl anhand einer IF-Abfrage eingeschaltet. Ist die letzte Farbe erreicht, so wird der Farbparameter wieder auf '0' gesetzt. Anschließend wird die Spaltenzahl um eins erhöht und der Cursor neu positioniert.

Die folgenden acht Zeilen beinhalten die eigentliche Ausgabe der Balken, wobei die vertikalen Balken von unten nach oben ausgegeben werden. Es werden so lange Reverse-Leerzeichen ausgegeben, bis der entsprechende Wert in dem Feld A() unter '8' sinkt. Dann wird anhand der IF-Abfragen als letztes Zeichen eines der Grafikzeichen ausgegeben, das mit seiner Zeilenzahl in der Zeichenmatrix mit dem Restwert in der Variablen A() übereinstimmt. Nach Ausdruck aller vierzig Balken wird eine Warteschleife durchlaufen und das Programm neu gestartet.

BALKENV 10 - 1010

Variablen:

Name	Typ	Bereich	Bedeutung
CD$	H	25 Zeichen nur	'Cursor nach unten'
CR$	H	40 Zeichen nur	'Cursor rechts'
F	H	0...8	Farbwert
I	H	1...6 / 1...40	Laufvariable
S	P	0...39	Spalte zur Cursor-positionierung
Z	.P	0...24	Zeile zur Cursor-positionierung

Felder (Arrays):				
Name	Dimen.	Typ	Bereich	Bedeutung
A	1..40	P	0...160	Balkenhöhe

Unterprogrammaufrufe :		
in	nach	Zweck
250	1000	Cursor positionieren
440	1000	Cursor positionieren

Verzweigungen nach außen :			
in Ze	nach	Bedingung	Bemerkung
keine			

2.3.2 Horizontale Balkendiagramme

Übungsziel:

- grafische Zeichen benutzen
- IF-Abfragen
- Farbgebung
- Balkendiagramme

```
10 POKE53280,0
20 POKE53281,0
30 PRINT"■
100 DIMA(24)
110 CR$="■"
120 CD$="■"
130 FORI=1TO6
140 CR$=CR$+CR$
150 CD$=CD$+CD$
160 NEXT
170 CR$=LEFT$(CR$,40)
180 CD$=LEFT$(CD$,5)
```

```
190 PRINT"■"
250 FORI=1TO24
260 REM A(I)=INT(RND(8)*320)
261 A(I)=I*14
270 NEXT
280 FORI=1TO24
281 F=F+1
282 IFF=1THENPRINT"▌
283 IFF=2THENPRINT"▐
284 IFF=3THENPRINT"▀
285 IFF=4THENPRINT"▄
286 IFF=5THENPRINT"▌
287 IFF=6THENPRINT"■"
288 IFF=7THENPRINT"▐"
289 IFF=8THENPRINT"▀":F=0
300 Z=I
310 GOSUB1000
320 IFA(I)>=8THENPRINT"■ ";:A(I)=A(I)-8:GOTO320
330 IFA(I)=7THENPRINT"■▌";
340 IFA(I)=6THENPRINT"■▐";
350 IFA(I)=5THENPRINT"■▀";
360 IFA(I)=4THENPRINT"▀ ";
370 IFA(I)=3THENPRINT"▀▌";
380 IFA(I)=2THENPRINT"▀▐";
390 IFA(I)=1THENPRINT"▀▀";
400 NEXT
410 FORI=1TO10000
420 NEXT
430 RUN
1000 PRINT"■";LEFT$(CR$,S);LEFT$(CD$,Z);
1010 RETURN
60000 SAVE"@:BALKENH",8
```

Ähnlich wie bei den vertikalen Balkendiagrammen verläuft das Programm für die horizontalen Balkendiagramme auch, so daß wir an dieser Stelle nur auf die Änderungen eingehen werden.

Zunächst ist die Anzahl der Balken statt auf 40 auf 24 zu begrenzen, sodaß dem Feld A() nur 24 Werte zugewiesen werden können. Dies hat Einfluß auf die Dimensionierung des Feldes, auf die FOR.. Next-schleife zum Besetzen des Feldes mit Zufallszahlen und auch auf die Anzahl der Balken, die ausgegeben werden.

Auch hier werden beim eigentlichen Ausdruck der Balken für jeweils acht Einheiten Reverse-Leerzeichen gedruckt und nur das letzte Zeichen des Balkendiagrammes besteht aus einem grafischen Zeichen. Im Gegensatz zu den vertikalen

Balkendiagrammen müssen hier jedoch anhand des Restwertes in A() die Spalten in der Zeichenmatrix von links abgezählt werden.

BALKENH **10 - 1010**

Variablen:

Name	Typ	Bereich	Bedeutung
CD$	H	25 Zeichen nur	'Cursor nach unten'
CR$	H	40 Zeichen nur	'Cursor rechts'
F	H	0...8	Farbwert
I	H	1...6 / 1...24	Laufvariable
S	P	0 (im Beispiel)	Spalte zur Cursorpositionierung
Z	P	0...24	Zeile zur Cursorpositionierung

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
A	1..24	P	0...320	Balkenlänge

Unterprogrammaufrufe :

in	nach	Zweck
310	1000	Cursor positionieren

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
keine			

2.3.3 Relativierung der Ausgabewerte

Übungsziel:

- Optimierung von Balkendiagrammen

```
100 Z=40
110 M=320
130 DIMA(Z)
140 GOSUB260
160 MA=0:FORI=1TOZ:IFMA<A(I)THENMA=A(I)
170 NEXT
190 DI=MA/M
210 FORI=1TOZ:A(I)=INT(A(I)/DI):NEXT
250 END
260 FORI=1TOZ
270 A(I)=INT(RND(0)*10000)
280 NEXT
290 RETURN
```

Dieses kurze Programmstück dient zum Anpassen der anfallenden Daten an die Bildschirmauflösung. Dies ist von Vorteil, wenn die anzuzeigenden Daten sehr groß oder sehr klein sind. Kleine Zahlen werden derart gespreizt, daß der größte Wert über die volle Bildschirmbreite bzw. Höhe geht und größere Daten werden derart zusammengezogen, daß ebenfalls der größte Wert auf die größte Bildschrimausdehnung gebracht wird.

Dies reicht aus, da in der Regel ja nicht die effektiven Zahlenwerte interessant sind, sondern nur das Verhältnis der Werte zueinander, was bei dieser Vorgehensweise bestehen bleibt.

In Zeile 100 wird die maximale Zahl der Balken angegeben und in Zeile 110 die maximale Auflösung. Wenn Sie das Programm im Zusammenhang mit dem Programm aus 2.3.1 oder 2.3.2 benutzen, so entfällt die Zeile 130. Bei vorgegebenen Zahlen ebenso die Zeile 140, die nur für den Testfall benötigt wird. Im weiteren wird zunächst das Maximum gesucht und dann der Divisor für alle Werte berechnet. Zum Abschluß werden noch alle Werte mit Hilfe des erhaltenen Divisors umgewandelt.

2.3.4 Frei programmierbare vertikale Balken

Übungsziel:

- frei handhabbare Balken
- programmieren mit zwei logischen Variablenebenen (normale Variablen und tätigkeitsbeschreibende Variablen)
- Farbumschaltung
- Cursorpositionierung

```
100 POKE53280,0
110 POKE53281,0
120 PRINT"■"
130 DATA24,12,90,4,5,6,20,5,4,5,70,5,2,7
140 DATA140,14,5,6,120,15,4,5,50,16,2,7
150 DATA130,24,5,6,150,25,4,5,90,26,2,7
160 DATA30,34,5,6,50,35,4,5,90,36,2,7
170 CR$="■"
180 CD$="■"
190 FORI=1TO6
200 CR$=CR$+CR$
210 CD$=CD$+CD$
220 NEXT
230 CR$=LEFT$(CR$,40)
240 CD$=LEFT$(CD$,25)
250 PRINT"■"
260 READZ    :REM INPUT"■GRUNDFARBE";Z
270 READBA   :REM INPUT"VIELFALT";BA
280 DIMH(BA)
290 DIMS(BA)
300 DIMF(BA)
310 DIMB(BA)
320 FORI=1TOBA
330 READH(I):REM INPUT"HOEHE";H(I)
340 READS(I):REM INPUT"=FALTE";S(I)
350 READB(I):REM INPUT"BREITE";B(I)
360 GOTO440
370 PRINT" 1 - ■WEISS
380 PRINT" 2 - ■HELLROT
390 PRINT" 3 - ■TUERKIS
```

```
400 PRINT" 4 - GRUEN
410 PRINT" 5 - GELB
420 PRINT" 6 - BLAU
430 PRINT" 7 - ORANGE
440 READF(I):REM INPUT"FARBE";F(I)
450 NEXT
460 PRINT""
470 FORI=1TOBA
480 S=S(I)
490 H=H(I)
500 B=B(I)
510 F=F(I)
520 GOSUB550
530 NEXT
540 GOTO780
550 IFF=1THENPRINT"
560 IFF=2THENPRINT"
570 IFF=3THENPRINT"
580 IFF=4THENPRINT"
590 IFF=5THENPRINT"
600 IFF=6THENPRINT"
610 IFF=7THENPRINT"
620 IFF=8THENPRINT"
630 GOSUB1000
640 A=H
650 IFA>=8THENPRINT"
660 IFA=7THENPRINT"
670 IFA=6THENPRINT"
680 IFA=5THENPRINT"
690 IFA=4THENPRINT"
700 IFA=3THENPRINT"
710 IFA=2THENPRINT"
720 IFA=1THENPRINT"
730 M=M+1:IFM=BTHENM=0:RETURN
740 IFM=BTHENM=0:RETURN
750 S=S+1:PRINT:GOTO630
760 PRINT:GOTO630
770 GOTO630
780 GETA$:IFA$=""THEN780
790 IFA$=""THEN780
800 RUN
1000 PRINT"";LEFT$(CR$,S);LEFT$(CD$,Z);
1010 RETURN
60000 SAVE"@:BALKENALL",8
```

Das Programm zum Setzen einzelner Balken mit verschiedener Breite und wahlweiser Spaltenpositionierung auf dem Bildschirm verläuft im Prinzip analog den in den Kapiteln

2.3.1 und 2.3.2 vorgestellten Programmen. Lediglich das Unterprogramm zum Drucken der Balken ist nun für einen einzigen Balken geschrieben. Aber gehen wir der Reihe nach:

In den Zeilen 130 bis 160 wurden einige DATA-Zeilen generiert, die die benötigten Daten für einen Testausdruck beinhalten. In der Regel wird man die Daten direkt über Bildschirm eingeben oder von einem größeren Programmkomplex zu Verfügung stellen und das vorliegende Programm nur als Unterprogramm aufrufen. Damit man die Möglichkeiten des Programmes besser austesten kann, ist es jedoch einfacher die benötigten Variablen in DATA-Zeilen zu schreiben, die jeweils leicht zu ändern sind. Dies wird im späteren Programm auch berücksichtigt.

Zunächst werden wie in den beiden anderen Programmen auch die beiden Variablen zum Positionieren des Cursors besetzt, dann wird die Grundzeile, auf der die Balken positioniert werden sollen, eingelesen. In der Bemerkung steht die komplette Zeile, falls die Daten am Bildschirm erfaßt werden sollen. Als nächstes wird die Anzahl der Balken, die ausgegeben werden sollen, eingelesen und auf Grund dieser Variablen (BA) werden die Felder zum Abspeichern der Höhe, der Spalte, der Farbe und der Breite der einzelnen Balken dimensioniert und in einer Schleife anschliessend die Variablen eingelesen. Auch hier steht in der Bemerkung wieder die komplette Zeile für die Erfassung vom Bildschirm. Die Zeilen 370 bis 430 werden in dieser Version übersprungen. Sie dienen zur Erfassung der gewünschten Farbe, für jeden einzelnen Balken am Bildschirm, wobei in diesem Menü der Farbbegriff jeweils auch die Farbe zugewiesen bekommt.

Dann wird der Bildschirm gelöscht, und in der folgenden Schleife wird der Ausdruck der einzelnen Balken veranlaßt, indem den einfachen Variablen S,H,B und F jeweils die Werte aus dem entsprechneden Feld für den aktuell zu druckenden Balken zugewiesen werden und das Unterprogramm zum Drucken des Balkens aufgerufen wird. Anschließend wird über den GET-Befehl gewartet, bis ein Tastendruck erfolgt, und das Programm wird neu gestartet.

In dem Unterprogramm zur Ausgabe eines einzelnen Balkens wird zunächst die Farbe vorbesetzt und daraufhin der Cursor entsprechend an der gewünschten Spalte positioniert. Der Hilfsvariablen A wird der Wert der Höhe (H) zugewiesen, und der Druck eines einzelnen Streifens erfolgt analog Kapitel 2.3.1. Dann wird der Merker für die Anzahl der ausgegebenen Streifen um eins erhöht. Ist bereits die gewünschte Breite des Balkens erreicht (M=B), wird der Mer-

ker zurückgesetzt und das Unterprogramm verlassen. Andernfalls wird der Cursor eine Spalte weiter positioniert und die erneute Ausgabe eines Streifens veranlaßt.

Änderungen und Ergänzungen

Nachdem bereits fast alle Werte eines Balkens variabel gehalten sind, dürfte es dem Leser nicht schwer fallen, auch noch die Grundzeile für jeden einzelnen Balken variabel zu halten. Versuchen Sie vielleicht auch mit aufsteigendem oder fallendem Z eine Fluchtlinie in Ihr Balkendiagramm zu bekommen. Vielleicht versuchen Sie auch bei allen in Kapitel 2.3 vorgestellten Programme alle sechzehn Farben zuzulassen.

Analog zu dem im letzten Kapitel vorgestellten Programmen kann auch versucht werden, horizontale Balkendiagramme in dieser flexiblen Art und Weise darzustellen.

BALKENALL 10 - 1010

Variablen:

Name	Typ	Bereich	Bedeutung
A	H	Integer	Hilfsvariable für Höhe des Balkens
A$	H	1 Zeichen	Warteschleife
B	P	Integer	Breite des auszugebenden Balkens
BA	H	Integer	Anzahl der Balken
CD$	H	25 Zeichen nur	'Cursor nach unten'
CR$	H	40 Zeichen nur	'Cursor rechts'
F	H	0...7	Farbe des auszugebenden Balkens
H	P	0...160	Höhe des auszugebenden Balkens
I	H	1...6 / 1...BA	Laufvariable
M	H	1..40	'Laufvariable' für die schon ausgegebene Breite des aktuellen Balkens

S	P	0...40	Spalte des auszugebenden Balkens
Z	P	0...24	Grundzeile zur Cursorpositionierung

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
H	1..BA	G	0...160	Höhe der Balken
S	1..BA	G	0...39	Spalte der Balk.
B	1..BA	G	0...39	Breite der Balk.
F	1..BA	G	0...8	Farbe der Balken

Unterprogrammaufrufe :

in	nach	Zweck
520	550	Einen Balken ausgeben
630	1000	Cursor positionieren

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
keine			

3

Sprites

3. Sprites

Die vom Commodore 64 gegebene Möglichkeit der Erstellung von Sprites, bietet eine sehr einfache Handhabe selbst Figuren zu definieren und diese - ohne Verwendung von Maschinenprogrammen - schnell auf dem Bildschirm zu bewegen. Besonders für Spiele ist es interessant, daß Kollisionen zwischen Sprites sehr schnell abgefragt werden können. Wir wollen im folgenden nach und nach ein kleines Spielprogramm entwickeln, bei dem wir alle Möglichkeiten der Sprites ausschöpfen wollen. Im Handbuch des Commodore 64 beschriebene Sachverhalte sollen hier nicht mehr erläutert werden, jedoch wollen wir die wichtigsten Sachen noch in einer Tabelle zusammenstellen.

3.1 Sprites definieren

Übungsziele:

- Hilfe bei der Erstellung von Sprites
- Entwicklung von kleineren Programmkonzepten ohne Verschachtelung
- Bildschirmhandhabung beim positionierten Einlesen
- Umrechnung von Bits
- POKE-Befehl

Will man mit Sprites arbeiten, so müssen diese zuerst definiert werden. Ein Sprite besteht aus einer Matrix von 24 x 21 Punkten (bei einfarbiger Darstellung, bei mehrfarbiger Darstellung aus 12 x 21 Punkten), die bitweise in 63 Byte abgelegt sind. Je drei aufeinander folgende Bytes ergeben eine Zeile eines Sprites. Da das im Handbuch beschriebene Errechnen der einzelnen Dezimalzahlen, die in die jeweiligen Bytes eingetragen werden müssen, sehr mühsam ist, wollen wir zu Anfang ein kurzes Programmstück erstellen, mit dem die Sprites sehr schnell erfasst werden können.

Zunächst wollen wir in dem Programm einen dunklen Hintergrund wählen mit

```
100 POKE 53281,11
```

und einen orangfarbenen Hintergrund mit

```
110 Poke 53280,8
```

dann müssen wir ein Feld dimensionieren in dem die Dezimalzahlen der einzelnen Bytes zwischengespeichert werden können mit

```
120 DIM BY(63)
```

wobei hier das BY für Byte steht. Anschließend wird mit

```
130 PRINT" (CLR/HOME) ";
```

der Bildschirm gelöscht. Mit der folgenden Schleife wird die Matrix für ein Sprite am Bildschirm mit lauter Punkten ausgegeben:

```
140 FOR I=1 TO 21
150 PRINT"........................"
160 NEXT
```

Wenn Sie jetzt das Programm starten, sehen Sie die stark vergrößerte Darstellung eines Sprites. In diese vorgegebenen Punkte wollen wir nun mit Reverse-Leerzeichen unsere Darstellung des Sprites einzeichnen.

Dazu müssen wir zuerst jedoch wieder die aktuelle Ausgabestelle für den Bildschirm in die linke obere Ecke setzen mit dem Befehl

```
170 PRINT" (HOME) ";
```

Das ';' ist wichtig, da sonst die Ausgabe bereits in der zweiten Zeile des Bildschirms erfolgen würde. Das Folgende mutet für einen unerfahrenen Programmierer vielleicht etwas undurchsichtig an, ist jedoch ganz einfach aufgebaut. Im Prinzip besteht das kurze Programmstück aus drei ineinander geschachtelten Schleifen. Die äußere Schleife wird 21mal durchlaufen, d.h. einmal für jede Zeile des Sprite. Die mittlere Schleife wird dreimal durchlaufen, weil je Zeile des Sprites 3 Byte benötigt werden. Der hauptsächliche Rechenvorgang spielt sich in der inneren Schleife ab, deren Anweisungen achtmal - einmal für jedes Bit des Bytes - durchlaufen wird. Wir erhalten also:

```
180 FOR I=1 TO 21
190 FOR J=1 TO 3
200 FOR K=1 TO 8
```

Zunächst müssen wir in der innersten Schleife, die im

Prinzip 504mal durchlaufen wird (einmal für jeden Punkt des Sprites), zunächst ein Zeichen von der Tastatur einlesen. Dies geschieht in Zeile

```
210 GET A$
```

Da der GET-Befehl nicht auf die Eingabe der Return-Taste wartet, sondern das Programm gleich fortsetzt, muß eine Schleife programmiert werden, die wartet, bis wirklich ein Zeichen über die Tastatur eingegeben wurde. Dies geschieht mit

```
220 IF A$="" THEN 210
```

Im Folgenden wollen wir festlegen, daß bei Eingabe eines "j" auf der Tastatur der Punkt in dem Sprite gesetzt werden soll und bei Eingabe irgendeiner anderen Taste kein Punkt gesetzt werden soll. Daraus resultiert:

```
230 IF A$= 'J' THEN SU = SU+2 (Pfeil nach oben) (8-k)
240 IF A$= 'J' THEN PRINT"(reverse on) (reverse off)";
```

Damit wird ein farbiges Leerzeichen, das sich vom Hintergrund abhebt, auf dem Bildschirm an der entsprechenden Stelle gesetzt. Im anderen Falle muß eingegeben werden

```
250 IF A$ NE "J" THEN PRINT " ";
```

womit der Punkt am Bildschirm gelöscht wird. Dann kann die innere Schleife beendet werden und der errechnete Wert für SU in dem vorher dimensionierten Feld BY(63) an der entsprechenden Stelle abgelegt werden. Diese Stelle errechnet sich aus der 'Zeilennummer - 1 * 3', womit die Bytes für volle Zeilen übersprungen werden, '+J' für die Nummer des Bytes in der aktuellen Zeile. Dann muß noch die Summe der errechneten Bits auf 0 gesetzt werden, und auch die mittlere Schleife kann beendet werden, was wie folgt aussieht:

```
260 NEXT
270 BY((I-1)*3+J) = SU
280 SU=0
290 NEXT
```

Da alle bisherigen PRINT-Befehle mit einem ';' abgeschlossen wurden, muß beim Abschluß einer Zeile (an dieser Stelle sind wir jetzt angelangt) noch ein PRINT-Befehl eingegeben werden um die Ausgabe des nächsten Zeichens am Beginn der nächsten Zeile zu veranlassen. Dann kann auch die äußere Schleife abgeschlossen werden.

```
300 PRINT
310 NEXT
```

Nachdem die Punkte des Sprites erfaßt wurden, wollen wir diese noch auf dem Bildschirm ausgeben. Es ist zwar auch eine Ausgabe auf dem Drucker möglich, aber nicht jeder hat einen Drucker, so daß wir uns hier auf den Bildschirm beschränken wollen. Da der Bildschirm weniger Zeilen als die Matrix Felder hat, müssen wir noch eine Warteschleife einbauen, mit der die Ausgabe unterbrochen werden kann; dazu nehmen wir ein bereits erprobtes Mittel: eine Warteschleife mit einem GET-Befehl.

Das Programm zum Ausgeben der errechneten Werte sieht dann wie folgt aus:

```
320 FOR J=1 TO 64
330 PRINT BY(J)
340 GET A$ : IF A$="" THEN 340
350 NEXT
```

Damit ist das komplette Listing fertig, und wir können das Programm mit RUN starten. Die einzelnen Dezimalzahlen die also in die entsprechenden Bytes für das Sprite einzutragen sind, werden von dem Programm ausgegeben. Diese kann man sich vom Bildschirm abschreiben und bei dem gewünschten Programm als DATA-Befehle eingeben. Es gibt jedoch auch eine andere Möglichkeit: die errechneten Werte automatisch in Programmzeilen mit Datas einzulesen. Dies erspart die Mühe des Abschreibens und Eintippens. Wozu hat man schließlich einen Computer ?!?

Dazu schreiben wir die erhaltenen Werte zunächst in den Arbeitsspeicher mit

```
360 INPUT "ANFANG BYTE-BLOCK";AF
370 HI = AF * 64
380 FOR I = 1 TO 63
390 POKE HI+I,BY(I)
400 NEXT
```

Warum der Eingabewert noch mit 64 multipliziert werden muß, sehen wir in den nächsten Kapiteln. Hier das Komplett-Listing :

```
100 POKE53281,11
110 POKE53280,6
120 DIMBY(64)
130 PRINT"■";
```

```
140 FORI=1TO21
150 PRINT"......................."
160 NEXT
170 PRINT"■";
180 FORI=1TO21
190 FORJ=1TO3
200 FORK=1TO8
210 GETA$
220 IFA$=""THEN210
225 IFA$="←"THENK=K-1:PRINT"■";:GOTO210
230 IFA$="J"THENSU=SU+2↑(8-K)
240 IFA$="J"THENPRINT"■ ■";
250 IFA$<>"J"THENPRINT"■ ■";
260 NEXT
270 BY((I-1)*3+J)=SU
280 SU=0
290 NEXT
300 PRINT
310 NEXT
320 FORJ=1TO64
330 PRINTBY(J)
340 GETA$:IFA$=""THEN340
350 NEXT
360 INPUT" ANFANG BYTE-BLOCK";AF
370 HI=AF*64
380 FORI=1TO63
390 POKE HI+I,BY(I)
400 NEXT
```

3.2 DATA-Anweisungen generieren

Obungsziel:

- Handhabung DATA-Statements
- Generieren von Programmzeilen
- Handhabung Tastaturpuffer

```
500 PRINT"■MACHE DATA-ZEILEN AUS SPEICHERINHALTEN"
510 INPUT"■STARTADRESSE IM SPEICHER";SA
520 INPUT"ANZAHL BYTES";AB
530 INPUT"STARTZEILE FUER DATA";SZ
540 IFSZ<300THEN120
550 PRINT"■";
```

```
560 FORZ=0TO(AB-1)/3
570 PRINTMID$(STR$(SZ+Z),2);"D♠";
580 FORI=0TO2
590 IF3*Z+I>=ABTHEN210
600 PRINTMID$(STR$(PEEK(SA+10*Z+I)),2);",";
610 NEXTI
620 PRINTCHR$(20)
630 POKE632+Z,13
640 NEXTZ
650 POKE631,19
660 POKE198,Z+1
670 END
```

Mit diesem Programm können DATA-Zeilen aus Speicherinhalten generiert werden, was besonders natürlich bei den Sprites von Vorteil ist, wenn diese schon im Speicher stehen. Zunächst wird die Startadresse im Speicher abgefragt, die die absolute Stelle im RAM angibt. In der Variablen AB wird die Anzahl der Bytes abgefragt, für die die DATA-Statements generiert werden sollen. Als letztes wird noch die Zeilennummer erfragt in der die DATA-Statements beginnen sollen. Die Startzeile für die Datas muß natürlich größer als 300 sein, da sonst das Programm selbst überschrieben wird.

In der folgenden Schleife werden alle Bytes in Dreiergruppen in DATA-Zeilen gepackt, so daß je Zeile eines Sprites auch eine DATA-Zeile vorhanden ist. Zunächst wird in Zeile 170 die Zeilennummer der DATA-Zeile und der Befehl DATA am Bildschirm ausgegeben. In einer weiteren Schleife werden dann die Speicherinhalte ausgelesen und als Dezimalzahl am Bildschirm durch Komma getrennt dargestellt. In Zeile 220 wird noch das letzte Komma in der DATA-Zeile eleminiert. Zeile 230 schreibt in den Tastaturpuffer eine '13' hinein, womit später - nach Abschluß des Programms - für jede DATA-Zeile ein RETURN ausgelöst wird, wodurch die Zeile dann in den Speicher übernommen wird. In Zeile 250 wird noch das erste Zeichen im Tastaturpuffer mit dem Befehl 'HOME' belegt und in Zeile 260 wird die Anzahl der gültigen Zeichen im Tastaturpuffer festgelegt.

Für andere Programme können wir uns also merken:

- Beginn des Tastaturpuffers in RAM-Adresse 631
- Anzahl der gültigen Zeichen im Tastaturpuffer: RAM-Adresse 198.

Wenn man per Programm irgendetwas an das Programm selbst übergeben will, muß man Zeilennummer und Programmzeile auf den Bildschirm bringen und diese über Tastaturpuffer mit

einem Carriage-Return (CHR$(13)) nach Programmende übernehmen lassen.

In dem vorliegenden Fall wird das Unterprogramm ja nur für Sprites benützt, so daß die Zeile 120 auch noch wegfallen kann und man die Variable AB konstant auf 63 setzen kann. Mit diesem Unterprogramm können aber auch Maschinenprogramme in DATA-Statements übernommen werden. Dazu muß dann jedoch die Anzahl der Bytes eingelesen werden, und es ist zu berücksichtigen, daß nur 21 Zeilen auf den Bildschirm passen, da auch noch die Systemmeldung 'ready.' untergebracht werden muß.

In diesem Falle kann man aber in Zeile 160 und 190 die '3' durch eine '10' ersetzen und in Zeile 180 die '2' durch '9', womit dann bis zu 10 Speicherzellen in eine DATA-Zeile übernommen werden.

3.3 Sprites erzeugen

Übungsziel:

- Spritedaten in RAM schreiben
- Sprites am Bildschirm sichtbar machen

Wir wollen im folgenden Programm zunächst das Sprite erzeugen und ihm dann definierte Bewegungen über Buchstaben der Tastatur zuordnen (siehe nächstes Kapitel). Dazu geben wir zunächst wieder die Farben für den Rahmen und den Hintergrund ein mit

```
100 POKE 53280,8
110 POKE 53281,11
```

Die DATA's wollen wir für die Sprites in den Zeilen 1000 bis 1999 ablegen und beginnen mit dem Sprite wie es vorher dargestellt wurde in Zeile 1000, wobei jeder DATA-Zeile 3 Zeilen des Sprite entsprechen, um spätere Korrekturen einfacher durchführen zu können.

```
1000 Data 0,24,0,0,24,0,0,60,0
1010 Data 0,60,0,0,60,0,0,126,0
1020 Data 0,126,0,0,126,0,12,126,48
1030 Data 12,255,48,13,255,176,15,129,176
1040 Data 15,165,240,15,129,240,31,255,248
1050 Data 63,255,252,127,255,254,255,255,255
1060 Data 0,231,0,0,231,0,0,231,0
```

Das Einlesen und Vorbesetzen der Video-Controller-Variablen für die einzelnen Sprites wollen wir in den Zeilen 2000 bis 2999 unterbringen. Zunächst müssen wir jedoch noch der Einfachheit halber die Startadresse des Video-Controllers (VIC) der Variablen V zuordnen und erhalten

```
120 V=53248
```

In Zeile 2000 wollen wir die gewünschten Bytes für die Anfangsbelegung aktivieren und erhalten somit, wenn wir dem 'Raumschiff' das Sprite 0 zuordnen

```
2000 POKE V+21,1
```

dann müssen wir dem Sprite noch den Speicherbereich zuweisen, wo seine Form vermerkt ist, wozu wir den ersten Teil des Kassettenrecorderpuffers hernehmen (Speicherzellen ab 832/13. Block).

```
2010 POKE 2042,13
```

In der folgenden Schleife werden die DATA's, die wir in den Zeilen ab 1000 definiert haben, in die entsprechenden Speicherzellen übertragen.

```
2020 FOR I=1 TO 63
2030 READ PU
2040 POKE 831+I,PU
2050 NEXT
```

Damit wir das Sprite auch auf dem Bildschirm sehen, müssen wir ihm noch seine X und Y Koordinaten zuordnen. Da es sich um Sprite 0 handelt ist das Register V für die X-Koordinate und das Register V+1 für die Y-Koordinate zuständig. Wir wollen das Sprite irgendwo links oben am Bildschirm erscheinen lassen und setzen deshalb X=50 und Y=50. Ebenso müssen wir noch die Farbe für das Sprite festlegen, wozu das Register V+39 herangezogen wird. Dort tragen wir den Code für Gelb (7) ein.

```
2052 X = 30
2054 Y = 50
2060 POKE V,50
2070 POKE V+1,50
2080 POKE V+39,7
```

Wenn wir nun das bisher eingetippte Programm mit RUN starten, erscheint das gewünschte Sprite links oben am Bildschirm.

3.4 Sprites bewegen

Übungsziel:

- Tastaturabfrage
- bewegen von Sprites
- automatische Bestimmung X/Y-Koordinaten

Um unser Sprite zu bewegen und dabei nicht auf den Zufall angewiesen zu sein,benutzen wir die Buchstaben der Tastatur, um die Richtung für das Sprite vorzugeben. Wer einen Joystick hat, fragt statt der Buchstaben die entsprechenden Bytes im Speicher für den Joystick ab. Wir wollen festlegen, daß das Sprite mit 'E' nach oben, mit 'D' nach rechts, mit 'S' nach links und mit 'X' nach unten geht (siehe Anordnung auf der Tastatur). Vorher wollen wir jedoch das Programmstück zum Unterbringen der neuen X/Y- Koordinaten in den entsprechenden Adressen des Video-Controll ers schreiben. Um nicht auf einen 'Illegal Quantity Error' zu laufen, müssen wir noch abprüfen, ob X und Y im zulässigen Bereich liegen. Das Programmstück sieht dann wie folgt aus:

```
4000 IF X LT 0 THEN X = 0: GOTO 4030
4010 IF X GT 300 THEN X = 255 : GOTO 4030
4020 POKE V,X
4030 IF Y LT 0 then Y = 0 : RETURN
4040 if Y GT 200 then Y = 200 : RETURN
4050 POKE V+1,Y
4060 RETURN
```

Für die Bewegung wollen wir zunächst eine Vorversion schreiben mit der die Bewegung einzeln beim jeweiligen Tastendruck erfolgt und anschließend das Programm so umgestalten, daß die Bewegung solange erfolgt, bis eine neue Taste gedrückt wird. Dazu brauchen wir wieder eine Schleife, die ein Zeichen von der Tastatur einliest und ein Programmblock der die eingelesenen Zeichen verarbeitet.

```
3000 GET A$
3010 IF A$ = '' THEN 3000
3020 IF A$ = 'E' THEN Y=Y-1 : GOSUB 4000 : GOTO 3000
3030 IF A$ = 'D' THEN X=X+1 : GOSUB 4000 : GOTO 3000
3040 IF A$ = "S" THEN X=X-1 : GOSUB 4000 : GOTO 3000
3050 IF A$ = "X" THEN Y=Y+1 : GOSUB 4000 : GOTO 3000
3060 GOTO 3000
```

Wenn wir nun das Programm mit RUN/RETURN starten, können wir nun mit den Tasten E,D,S und X das Sprite nach oben, rechts, links und unten bewegen. Damit das Sprite etwas größer erscheint können wir das Programm mit RUN/STOP unterbrechen und im Direkt-Modus mit POKE V+29,1 und POKE V+23,1 das Sprite auf das 4-fache vergrößern.

Da es recht mühsam ist, das Sprite immer nur minimal mit einem Tastendruck zu verschieben, wollen wir nun das Programm so umstellen, daß das Sprite sich solange in die gewünschte Richtung bewegt, bis eine andere Taste gedrückt wird. Dies erreichen wir mit:

```
3000 Print "CLR/HOME"
3010 GET A$
3020 IF A$ = '' THEN 3040
3030 B$ = A$
3040 IF B$ = "E" THEN Y = Y-1 : GOSUB 4000 : GOTO 3010
3050 IF B$ = "D" THEN X = X+1 : GOSUB 4000 : GOTO 3010
3060 IF B$ = "S" THEN X = X-1 : GOSUB 4000 : GOTO 3010
3070 IF B$ = "X" THEN Y = Y+1 : GOSUB 4000 : GOTO 3010
3080 GOTO 3000
```

Durch das Übergehen der Zeile 3020 in Zeile 3040 wird sichergestellt, daß die Bewegung auch ausgeführt wird, wenn keine Taste gedrückt wurde. Wurde ein anderes Zeichen gedrückt, wo wird dies in der Zeile 3030 der Variablen B$ zugeordnet und, sofern ein gültiges Zeichen vorlag, dies in die Bewegung übernommen. Bei allen anderen Zeichen hält die Bewegung an. Damit sind jedoch noch nicht alle Bewegungsmöglichkeiten ausgereizt, es fehlen noch die Diagonalen. Wir wollen also verwenden:

R - rechts oben
W - links oben
C - rechts unten
Z - links unten

(jeweils analog der Anordnung auf der Tastatur) und müssen unser Programm wie folgt ändern:

```
3080 IF B$ = 'R' THEN Y = Y-1:X=X+1:GOSUB 4000: GOTO 3010
3090 IF B$ = 'W' THEN Y = Y-1:X=X-1:GOSUB 4000: GOTO 3010
3100 IF B$ = 'C' THEN Y = Y+1:X=X+1:GOSUB 4000: GOTO 3010
3110 IF B$ = 'Z' THEN Y = Y+1:X=X-1:GOSUB 4000: GOTO 3010
```

Wenn wir jetzt das Programm starten, haben wir schon acht Bewegungsrichtungen für unseren 'Abfangjäger', wobei bei einem Anstoß an den Rand bei den diagonalen Bewegungsrichtungen jeweils nur noch eine Richtung ausgeführt wird. Bei geraden Bewegungsrichtungen am Rand und bei diagonalen Be-

wegungsrichtungen in der Ecke bleibt unser Sprite stehen.

Nun wollen wir uns noch ein zweites Sprite definieren wie es in den Zeilen ab 1100 dargestellt ist:

```
1100 DATA 12, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0
1110 DATA 12, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0
1120 DATA 12, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0
1130 DATA 12, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0
1140 DATA 12, 0, 48, 0, 0, 0, 0, 0, 0, 0, 0
1150 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0
1160 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0
```

Wie man leicht sieht besteht dieses Sprite nur aus wenigen Punkten und ist so konstruiert, daß es überlagert mit unserem 'Abfangjäger' Schüsse aus dessen 'Kanone' simuliert. Für diese Schüsse wollen wir Sprite Nr.1 heranziehen und müssen dementsprechend ändern

```
2000 POKE V+21,3
```

und einfügen:

```
2100 POKE 2041,14
```

womit wir dem Sprite 1 den 14. Datenblock zuordnen, d.h. die Speicheradressen ab 896. Für die Zuordnung ergeben sich folgende Zeilen:

```
2110 FOR I=1 TO 63
2120 READ PU
2130 POKE 895+1,PU
2140 NEXT
2150 X1 = 100
2160 Y1 = 100
```

Für die X/Y-Koordinaten und die Farbe ergibt sich

```
2170 POKE V+2,X
2180 POKE V+3,Y
2190 POKE V+40,5
```

wenn die Farbe grün sein soll und Sprite 1 mit Sprite 0 deckungsgleich sein soll.

Da das Sprite Nr. 1 jedoch nur erscheinen soll, wenn geschossen wird, ändern wir Zeile 2000 wieder ab in

```
2000 POKE V+21,1
```

und ergänzen

```
3120 IF B$="(Pfeil nach oben)" THEN GOSUB 5000 : GOTO 3010
```

In dem Unterprogramm ab 5000 müssen wir nun das Sprite für den Schuß einschalten

```
5000 POKE V+21,3
```

sowie einen Merker setzen, daß der Schuß im Moment 'mitspielt' :

```
5010 S=1
```

dann müssen wir die aktuellen X/Y-Koordinaten dem Schußsprite zuordnen mit

```
5020 X1 = X
5030 Y1 = Y
5040 POKE V+2,X1
5050 POKE V+3,Y1
```

wonach wir das Unterprogramm mit

```
5060 RETURN
```

verlassen können. In der Schleife die das Raumschiff bewegt, muß nun auch die Bewegung für das Schuß-Sprite durchgeführt werden. Wir verlegen also Zeile 4000 nach Zeile 4005 und rufen das zu schreibende Unterprogramm ab Zeile 5100 aus Rechenzeitgründen nur dann auf, wenn wirklich ein Schuß gestartet wurde.

```
4000 IF S THEN GOSUB 5100
```

In dem Unterprogramm ab 5100 brauchen wir nur von der Y-Koordinate des Sprites 1 '1' abzuziehen,

```
5100 Y1 = Y1 - 1
```

und den so erhaltenen Wert dem entsprechenden Register (3) zuzuordnen mit

```
5110 POKE V+3,Y1
```

damit können wir das Unterprogramm mit

```
5120 RETURN
```

verlassen.

3.5 Kollision von Sprites

Übungsziel:

- Kollision von unterschiedlichen Sprites bestimmen
- Sprites per Zufall auf dem Bildschirm bewegen
- Farbwechsel
- schrittweise Programmentwicklung

Sehr interessant bei dem Commodore 64 ist die Möglichkeit, festzustellen, ob sich Sprites überlagern. Dies geschieht mittels eines Registers, wo aus den gesetzten Bits abgelesen werden kann, welche Sprites miteinander kollidieren.

Unser schon bekanntes Spiel wollen wir nun ergänzen, indem wir auch noch einen feindlichen Raumgleiter (Star Wars läßt grüßen) als drittes Sprite definieren. Die DATA's dazu befinden sich in den Zeilen 1200 bis 1260 und lauten wie folgt:

```
1200 DATA 192, 0, 3, 192, 0, 3, 192, 0, 3
1210 DATA 192, 0, 3, 192, 0, 3, 192, 0, 3
1220 DATA 192, 0, 3, 192, 24, 3, 192, 60, 3
1230 DATA 192, 102, 3, 255, 231, 255, 192, 102, 3
1240 DATA 192, 60, 3, 192, 24, 3, 192, 0, 3
1250 DATA 192, 0, 3, 192, 0, 3, 192, 0, 3
1260 DATA 192, 0, 3, 192, 0, 3, 192, 0, 3
```

und das schon bekannte Vorbesetzen der Werte des Sprites sieht dann wie folgt aus:

```
2200 POKE 2042,15
2210 FOR I=1 TO 63
2220 READ PU
2230 POKE 959+I,PU
2235 NEXT
2240 X2 = 200 * RND(-TI)+100
2250 Y2 = 150 * RND(-TI)+100
2260 POKE V+4,X2
2270 POKE V+5,Y2
2280 POKE V+41,1
```

Wie man an den Zeilen 2240 und 2250 sehen kann, wird das Sprite irgendwo am Bildschirm dargestellt.

Auch sollen die Bewegungen des Sprites automatisch erfolgen, wobei unser 'Abfangjäger' natürlich schneller sein soll. Wir wollen festlegen, daß Sprite 2 jedes 5.mal bewegt werden soll, wenn eine Bewegung für Sprite 0 geprüft wurde und geben deshalb noch die Zeile

```
3014 ZA = ZA + 1 : IF ZA GT 5 THEN GOSUB 6000
```

In dem Unterprogramm ab Zeile 6000 wollen wir dann das Sprite 2 automatisch bewegen, was dann wie folgt aussieht:

```
6000 X2 = X2 + 5 * RND(5) -3
6005 IF X2 LT 0 OR X2 GT 255 THEN 6099
6010 Y2 = Y2 + 5 * RND(5) -2
6015 IF Y2 LT 0 OR Y2 GT 255 THEN 6099
6020 POKE V+4,X2
6030 POKE V+5,Y2
6040 ZA=0
6099 RETURN
```

Der Zähler (ZA) für die Tastaturabfrage wird gleich in dem Unterprogramm wieder auf 0 zurückgesetzt. Wie wir in Zeile 2260 und 2270 schon gesehen haben sind die Adressen für die X/Y-Koordinaten den Adressen 4 + 5 des Videochips zugeordnet.

Jetzt sind wir bereits in der Lage unseren 'Abfangjäger' mittels der Buchstabentasten zu bewegen, mittels der Pfeiltaste einen Schuß loszulassen und haben ein drittes Sprite, was sich willkürlich am Bildschrm bewegt. Wenn Sie jetzt etwas 'testen', werden Sie feststellen, daß sich bei Kollisionen der Sprites natürlich nichts tut. Die Schüsse durchdringen einfach den feindlichen Raumgleiter.

Das Register zur Feststellung für die Kollisionen befindet sich im Videochip in der Adresse 30, und diese wollen wir nun abfragen mit

```
3016 IF PEEK (V+30) NE 0 THEN GOSUB 7000
```

In dem Unterprogramm ab Zeile 7000 besetzen wir zunächst die Hilfsvariable K0 mit dem Ergebnis des Sprite-Kollision-Bytes und prüfen anschließend ab, was vorliegt. Interessant sind für uns nur die Fälle K0=3, 6 und 5, da diese nur auftreten können. Bei K0=3 liegt eine Kollison zwischen dem Schuß und dem Raumgleiter vor, die natürlich auch beim Abschuß entsteht. Diese Kollision darf logischerweise im weiteren Spielverlauf nicht ausgewertet werden und wir verlassen das Unterprogramm sofort wieder. Aus Rechenzeitgründen wurde unter anderem diese Abfrage nach vorne gelegt, da das Programm leider schon zu langsam ist.

Bei einem KO=6 liegt eine Kollision zwischen dem Schuß und dem feindlichen Raumgleiter vor und wir verzweigen in ein entsprechendes Unterprogramm und bei einem KO=5 liegt eine Kollision zwischen den beiden Raumschiffen vor.

Es ergeben sich also die Zeilen

```
7000 KO = PEEK(V+30)
7005 IF KO=3 THEN RETURN
7010 IF KO=6 THEN7100
7020 IF KO=5 THEN7200
7030 RETURN
```

Um die Farbmöglichkeiten des Commodore 64 nochmals darzustellen, wurde bei dem 'Abschuß' von Sprite 2 nur eine Schleife programmiert, in der alle Farben des Commodore 64 vorliegen. Zunächst wird jedoch das Sprite 1 (Schuß) abgeschaltet, da es seine Schuldigkeit getan hat. Die Schleife, die durch die Variable WA durchlaufen wird, dient nur zum Verlangsamen des Farbwechsels. Deshalb ist es auch unerheblich, daß jedesmal wieder der Schuß innerhalb der Schleife abgeschaltet wird.

Als Abschluß dieses Unterprogramms wird auch noch Sprite 2 ausgeschaltet (durch Einschalten von ausschließlich Sprite 0) und der Merker für einen gesetzten Schuß wird ausgeschaltet.

```
7100 FOR F=0 TO 15 : FOR WA=1 TO 15
7105 POKE V+21,5
7110 POKE V+41,F
7120 NEXT : NEXT
7130 POKE V+21,1
7140 S=0
7150 RETURN
```

Ähnliches wie im letzten Unterprogramm wird auch bei einer Kollision der Raumschiffe programmiert, jedoch lassen wir die Nummern der Farben der beiden Raumschiffe gegeneinander laufen, während ihres Farbwechsels. Ein Schuß bleibt natürlich weiterhin bestehen. Als Abschluß wird wieder die Farbe des Raumgleiters mit dem ursprünglichen Wert besetzt und der Raumgleiter selbst um je 50 Punkte in XY-Richtung weiterbewegt, damit es im weiteren Programmablauf nicht sofort wieder zu einer Kollision kommt.

Damit wäre eigentlich ein Rahmen für ein Spiel gelegt. Zwei Kleinigkeiten wollen wir jedoch noch ergänzen. Zum einen wollen wir den Schuß ausschalten, sobald er am oberen Rand erscheint, dies geschieht mit

```
5200 S=0
5210 POKE V+21,5
5220 RETURN
```

und zum anderen wollen wir natürlich nicht jedesmal das Programm neu mit RUN starten, sondern geben auch noch die Zeile

```
3130 IF B$ = "N" THEN POKE V+21,5 : GOTO3010
```

ein, womit durch das Drücken der Taste 'N' ein neues Sprite 2 erzeugt wird.

3.6 Sontiges / Zusammenfassung

Ziel von Kapitel 3 war es nicht, ein Superspiel zu programmieren, sondern Ihnen die Möglichkeiten der Programmierung von Sprites aufzuzeigen. Daß das Spiel sehr langsam geht, liegt an dem Basic-Interpreter. Interessierte Leser können dies sicherlich mit Hilfe des in Kapitel 5 beschriebenen Assemblers verschnellern. Dies würde jedoch den Rahmen dieses Buches sprengen. In Band 3 werden wir auf dieses Thema noch einmal eingehen.

Um Ihnen die weiteren Änderungen des vorliegenden Programmes zu vereinfachen, haben wir im Anschluß nochmal das komplette Listing abgedruckt und eine Übersicht über die verwendeten Variablen, sowie deren Funktion.

Dieses Spiel bietet dem Spiele-Fan ein weites Betätigungsfeld. Es können z.B. weitere Sprites definiert werden z.B. der 'Abfangjäger' in vier verschiedenen Drehrichtungen, dementsprechend müssen auch die Schüsse waagerecht, senkrecht und jeweils in beiden Richtungen verlaufen können. Weiterhin könnte man ein Zeitlimit festsetzen und eine Punktvergabe für getroffene gegnerische Raumschiffe einbauen. Da auch eine Kollision zwischen Sprite und Hintergrund abgefragt werden kann, kann man mit grafischen Zeichen eine Basis aufbauen, an die man den 'Abfangjäger' andocken lassen kann. Wie Sie sehen, gibt es hier schier unerschöpfliche Möglichkeiten.

```
100 REM ****************************************
101 REM *       R A K E T E N S P R I T E S    *
102 REM ****************************************
110 POKE53280,8
120 POKE53281,11
130 V=53248
9?? REM ****************************************
```

```
998 REM *   DATAS FUER SPRITE 0                  *
999 REM ********************************************
1000 DATA0,24,0,0,24,0,0,60,0
1010 DATA0,60,0,0,60,0,0,126,0
1020 DATA0,126,0,0,126,0,12,126,48
1030 DATA12,255,48,13,255,176,15,129,176
1040 DATA15,165,240,15,129,240,31,255,248
1050 DATA63,255,252,63,255,252,63,255,252
1060 DATA0,231,0,0,231,0,0,231,0
1097 REM ********************************************
1098 REM *   DATAS FUER SPRITE 1                    *
1099 REM ********************************************
1100 DATA12,0,48,0,0,0,0,0,0
1110 DATA12,0,48,0,0,0,0,0,0
1120 DATA12,0,48,0,0,0,0,0,0
1130 DATA12,0,48,0,0,0,0,0,0
1140 DATA12,0,48,0,0,0,0,0,0
1150 DATA0,0,0,0,0,0,0,0,0
1160 DATA0,0,0,0,0,0,0,0,0
1197 REM ********************************************
1198 REM *   DATAS FUER SPRITE 2                    *
1199 REM ********************************************
1200 DATA192,0,3,192,0,3,192,0,3
1210 DATA192,0,3,192,0,3,192,0,3
1220 DATA192,0,3,192,24,3,192,60,3
1230 DATA192,102,3,255,231,255,192,102,3
1240 DATA192,60,3,192,24,3,192,0,3
1250 DATA192,0,3,192,0,3,192,0,3
1260 DATA192,0,3,192,0,3,192,0,3
1997 REM ********************************************
1998 REM *   SPRITE 0 VORBESETZEN                   *
1999 REM ********************************************
2000 POKEV+21,5
2010 POKE2040,13
2020 FORI=1TO63
2030 READPU
2040 POKE831+I,PU
2050 NEXT
2052 X=30
2054 Y=50
2060 POKEV,X
2070 POKEV+1,Y
2080 POKEV+39,7
2097 REM ********************************************
2098 REM *   SPRITE 1 VORBESETZEN                   *
2099 REM ********************************************
2100 POKE2041,14
2110 FORI=1TO63
2120 READPU
2130 POKE895+I,PU
2135 NEXT
```

```
2140 X1=100
2150 Y1=100
2160 POKEV+2,X1
2170 POKEV+3,Y1
2180 POKEV+40,5
2197 REM ******************************************
2198 REM *   SPRITE 2 VORBESETZEN                 *
2199 REM ******************************************
2200 POKE2042,15
2210 FORI=1TO63
2220 READPU
2230 POKE959+I,PU
2235 NEXT
2240 X2=200*RND(-TI)+100
2250 Y2=150*RND(-TI)+100
2260 POKEV+4,X2
2270 POKEV+5,Y2
2280 POKEV+41,1
2997 REM ******************************************
2998 REM *   BEWEGUNGSABLAUF                      *
2999 REM ******************************************
3000 PRINT"■"
3010 GETA$
3012 IFSTHENGOSUB5100
3014 ZA=ZA+1:IFZA>5THENGOSUB6000
3016 IFPEEK(V+30)<>0THENGOSUB7000
3020 IFA$=""THEN3040
3022 C$=B$
3030 IFB$<>A$THENB$=A$
3040 IFB$="E"THENY=Y-1:GOSUB4000:GOTO3010:POKEV+21,7
3050 IFB$="D"THENX=X+1:GOSUB4000:GOTO3010
3060 IFB$="S"THENX=X-1:GOSUB4000:GOTO3010
3070 IFB$="X"THENY=Y+1:GOSUB4000:GOTO3010
3080 IFB$="R"THENY=Y-1:X=X+1:GOSUB4000:GOTO3010
3090 IFB$="W"THENY=Y-1:X=X-1:GOSUB4000:GOTO3010
3100 IFB$="C"THENY=Y+1:X=X+1:GOSUB4000:GOTO3010
3110 IFB$="Z"THENY=Y+1:X=X-1:GOSUB4000:GOTO3010
3115 IFSTHEN3010
3120 IFB$="↑"THENGOSUB5000:B$=C$:GOTO3010
3130 IFB$="N"THENPOKEV+21,5:GOTO3010
3997 REM ******************************************
3998 REM *   SPRITE 0 BEWEGEN                     *
3999 REM ******************************************
4000 IFSTHENGOSUB5100
4005 IFX<0THENX=0:GOTO4030
4010 IFX>255THENX=255:GOTO4030
4020 POKEV,X
4030 IFY<0THENY=0:RETURN
4040 IFY>200THENY=200:RETURN
4050 POKEV+1,Y
4060 RETURN
```

```
4997 REM ********************************************
4998 REM *   SPRITE 1 BEWEGEN                       *
4999 REM ********************************************
5000 POKEV+21,7
5010 S=1
5020 X1=X
5030 Y1=Y
5040 POKEV+2,X1
5050 POKEV+3,Y1
5060 RETURN
5100 Y1=Y1-1
5102 IFY1<0THEN5200
5110 POKEV+3,Y1
5120 RETURN
5200 S=0
5210 POKEV+21,5
5220 RETURN
5997 REM ********************************************
5998 REM *   SPRITE 2 BEWEGEN                       *
5999 REM ********************************************
6000 X2=X2+5*RND(5)-3
6005 IFX2<0ORX2>255THEN6005
6010 Y2=Y2+5*RND(5)-2
6015 IFY2<0ORY2>255THEN6005
6020 POKEV+4,X2
6030 POKEV+5,Y2
6040 ZA=0
6099 RETURN
6997 REM ********************************************
6998 REM *   KOLLISIONSPRUEFUNG                     *
6999 REM ********************************************
7000 KO=PEEK(V+30)
7005 IFKO=3THENRETURN
7010 IFKO=6THEN7100
7020 IFKO=5THEN7200
7030 RETURN
7097 REM ********************************************
7098 REM *   SPRITE 1 GETROFFEN                     *
7099 REM ********************************************
7100 FORF=0TO15:FORWA=1TO15
7105 POKEV+21,5
7110 POKEV+41,F
7120 NEXT:NEXT
7130 POKEV+21,1
7140 S=0
7150 RETURN
7197 REM ********************************************
7198 REM *   SPRITE 2 GETROFFEN                     *
7199 REM ********************************************
7200 FORF=0TO15:FORWA=1TO15
7210 POKEV+41,F
```

```
7220 POKEV+39,15-F
7230 NEXT:NEXT
7235 POKEV+39,7
7237 X=X+50
7238 Y=Y+50
7240 RETURN
```

RAKETE **10 - 7240**

Variablen:

Name	Typ	Bereich	Bedeutung
A$	G	1 Zeichen	Einlesen von Tastatur
B$	G	1 Zeichen	'Bewegungszeichen'
C$	G	1 Zeichen	Zwischenspeichern des Bewegungszeichens
F	H	0...15	Farbe für Farbwechsel
KO	H	0...255	Kollisionsmerker
PU	H	0...255	HV zum Umsetzen der DATA's für die Sprites
S	E/R	0/1	Flag für Schuss
V	G	Konstante 53248	Adresse VIC
WA	h	1...15	Warteschleife
X	G	0...319	X-Koordinate Sprite 0
X1	G	0...319	X-Koordinate Sprite 1
X2	G	0...319	X-Koordinate Sprite 2
Y	G	0...199	Y-Koordinate Sprite 0
Y1	G	0...199	Y-Koordinate Sprite 1
Y2	G	0...199	Y-Koordinate Sprite 2
ZA	G	0...5	Zähler für Bewegung Sprite 2

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
keine				

Unterprogrammaufrufe :

in	nach	Zweck
3014	6000	Sprite 2 bewegen
3016	7000	Kollision prüfen
3040	4000	Sprite 0 bewegen
3050	4000	Sprite 0 bewegen
3060	4000	Sprite 0 bewegen
3070	4000	Sprite 0 bewegen
3080	4000	Sprite 0 bewegen
3090	4000	Sprite 0 bewegen
3100	4000	Sprite 0 bewegen
3110	4000	Sprite 0 bewegen
3120	5000	Schuss einschalten
4000	5100	Schuss bewegen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
keine			

4

Die hochauflösende Grafik

4. Hochauflösende Grafik

Eine der beiden hervorstechenden Merkmale des Commodore 64 gegenüber den Geräten der Serie 2000 bis 8000 ist die hochauflösende Grafik von waagrecht 320 Punkten und senkrecht 200 Punkten, also insgesamt 64000 Punkten. Um sich zu veranschaulichen wie die hochauflösende Grafik funktioniert, wollen wir eine andere kleine Rechnung aufstellen: Jedes Zeichen auf dem Bildschirm wird aus einer Punktmatrix von 8 x 8 Punkten (=64) gebildet. Da 1000 Zeichen auf den Bildschirm passen, ergibt sich auch so wieder eine Einzelpunktanzahl für den normalen Textteil von 64000 Punkten. Diesen Zusammenhang werden wir später an verschiedenen Stellen noch benutzen müssen.

4.1 Allgemeines

4.1.1 Aufbau der Grafikspeicher

Eines sei gleich vorweg gesagt: hochauflösende Grafik in Zusammenhang mit dem Textmodus normal ist nicht möglich. Gesteuert wird die hochauflösende Grafik von dem Videochip (VIC), dessen Adresse wir von den Sprites her schon kennen: 53248. Ebenso wie dem Computer gesagt werden muß, das eine oder andere Sprite einzuschalten, muß man dem Computer auch mitteilen, daß er mit hochauflösender Grafik arbeiten soll. Dazu ist in der Adresse 53248+17 (im folgende kurz V+17 genannt) das Bit 32 zu setzen, was mit dem Befehl

```
POKE V+17,59          (Normaler Inhalt: 27)
```

geschieht. Außerdem sind in dem Byte V+24 noch das Bits 1 und zwei zu setzen, mit dem Befehl:

```
POKE V+24,24          (Normaler Inhalt: 21).
```

Jetzt hat man zwar den Rechner auf Grafik umgeschaltet, jedoch wird - wenn man die Befehle alleine eingibt - nur wirres Zeug am Bildschirm erscheinen. Zwei Sachen müssen vorher noch durchgeführt werden: Die Farbbestimmung für die Grafik und den Grafikspeicher löschen.

Der Commodore 64 kennt im Grafikmodus im Prinzip zwei Zustände. Zum einen das Zeichnen mit einer Farbe auf einem anderen farbigen Hintergrund und zum anderen ein Multi-Color-Modus, bei dem bis zu vier Farben verwendet werden können. Auf den Multi-Color-Modus - der für den Anfänger etwas schwierig zu handhaben ist - wollen wir in diesem Buch verzichten und in Band 3 der Reihe 'Das Commodore 64-Buch' näher eingehen.

Im RAM liegt in den Zellen 1024 bis 2023 der Bildschirmspeicher für den Textmodus. Wie wir an der anfangs erwähnten Rechnung schon gesehen haben, werden auch im Prinzip hier 64000 Punkte abgelegt. Der Unterschied zu dem weiter unten erklärten Farbspeicher besteht darin, daß der Bildschirmspeicher für je 64 Punkte ein Byte (=8 Bits) enthält das angibt, welches Zeichen aus dem Zeichengenerator zu holen ist. Dieses Zeichen füllt dann die 64 Punkte als Ganzes aus. Die 1 KByte Speicher für den Bildschirm sind also nur für maximal 8000 Punkte zu gebrauchen. Oder man nimmt sie zur Speicherung der Vorder- und Hintergrundfarbe je Zeichen her. Dazu werden in einem Byte (=1 Zeichen) des Bildschirmspeichers zwei Halbbyte zu je vier Bit gespeichert, von denen die ersten vier Bit die Punktfarbe und die letzten vier Bit die Hintergrundfarbe angeben. Mit ein bischen Binär-Arithmetik stellt man sehr leicht fest, daß die sechzehn Farben des Commodore komplett in vier Bit darstellbar sind.

Wenn man also mit folgender Programmschleife

```
FOR I=1024 TO 2023
POKE I,HF+16*PF
NEXT I
```

eine in den Variablen HF und ZF festgelegte Hintergrund- bzw. Zeichenfarbe in den Bildschirmspeicher schreibt, ist das Zufallsbild von vorhin nur noch zweifarbig. Die Speicherung der Farbcodes für je 64 Punkte ermöglicht es später Teilbereiche des Bildschirms bis hin zu einer minimalen Auflösung von 8 x 8 Punkten jeweils andersfarbig darzustellen. Man könnte also nach Belieben jedem Zeichen des Textspeichers eine andere Zeichen- und Hintergrundfarbe zuordnen, wie es auch vom Rechner selbst im Textmodus gemacht wird. Dies erkennt man am Schreiben eines Textes und zwischenzeitliches Umschalten der Farbe mit den CTRL- bzw. Commodore-Taste sehr leicht.

Wenn man die Speicherzellen bildschirmbezogen zugeordnet durchnumeriert, ergibt die Relation 'Stelle am Bildschirm / Nummer im RAM folgende Konstellation:

```
  1   2   3   4   5   6   7   8   9 ...   38  39  40
 41  42  43  44  45  46  47  48  49 ...   78  79  80
 81  82  83  84  85  86  87  88  88 ...  118 119 120
  .   .   .   .   .   .   .   .   . ...    .   .   .
  .   .   .   .   .   .   .   .   . ...    .   .   .
  .   .   .   .   .   .   .   .   . ...    .   .   .
```

Die Information, welcher Punkt bei einer hochauflösenden Grafik am Bildschirm gesetzt werden soll oder nicht, wird in einem anderen Speicherbereich verwaltet. Dies sind die RAM-Adressen 8192 bis 16383. Wie man leicht sieht: 8 Kilo-Byte.

Dabei wird jeweils einem Bit ein Punkt auf dem Bildschirm zugeordnet. Für ein Bildschirmzeichen werden also 8 Byte benötigt. Diese 8 Byte werden auch aus den vorher beschriebenen 8 KByte fortlaufend herausgenommen, so daß sich für die Punkte folgendes Muster ergibt:

In Klammern jeweils ein Byte

```
(    1    2    3    4    5    6    7    8) (   65   66 ...
(    9   10   11   12   13   14   15   16) (   73   74 ...
(   17   18   19   20   21   22   23   24) (   81   82 ...
(   25   26   27   28   29   30   31   32)      .
(   33   34   35   36   37   38   39   40)      .
(   41   42   43   44   45   46   47   48)      .
(   49   50   51   52   53   54   55   56)      .
(   57   58   59   60   61   62   63   64)      .

( 2561 2562 2563 2564 2565 2566 2567 2568)  ( 2625 ...
( 2569 2570 2571 2572 2573 2574 2575 2576)  ( 2633 ...
     .    .    .    .    .    .    .    .         .
     .    .    .    .    .    .    .    .         .
```

oder in Byte (analog zur Anordnung im RAM):

```
  1    9   17   25   33   41   49 ...
  2   10   18   26   34   42   50 ...
  3   11   19   27   35   43   51 ...
  .    .    .    .    .    .    . ...
  .    .    .    .    .    .    . ...
  8   16   24   32   40   48   56 ...

321  329  337 ...
322  330  338 ...
  .    .    . ...
  .    .    . ...
328  336  344 ...
```

```
                                641                    ...
     .
     .
```

Dabei muß man hierbei zum Setzen der Bits rechnerisch wie in allen anderen Fällen (vgl. Sprites) auch vorgehen.

```
!=========================================================!
!                                                         !
!    GRAFIK EINSCHALTEN                 20000 - 20110     !
!                                                         !
!=========================================================!
!                                                         !
! Variablen:                                              !
!                                                         !
!---------------------------------------------------------!
! Name ! Typ  ! Bereich          ! Bedeutung              !
!---------------------------------------------------------!
! HF   !  G   ! 0...15           ! Hintergrundfarbe       !
! LV   !  H   ! 1024...2023 /    ! Laufvariable           !
!      !      ! 8192...16383     !                        !
! V    !  G   ! konstant 53248   ! Adresse VIC            !
! ZF   !  G   ! 0...15           ! Zeichenfarbe (Punkte)  !
!=========================================================!
!                                                         !
! Unterprogrammaufrufe :                                  !
!                                                         !
!---------------------------------------------------------!
! in   ! nach  ! Zweck                                    !
!---------------------------------------------------------!
! keine!       !                                          !
!=========================================================!
!                                                         !
! Verzweigungen nach außen :                              !
!                                                         !
!---------------------------------------------------------!
! in Ze ! nach  ! Bedingung         ! Bemerkung           !
!---------------------------------------------------------!
! 20110 ! RETURN! Normales Ende     !                     !
!=========================================================!
```

4.1.2 Punkt setzen

```
20400 REM ****************************************
20401 REM *    P U N K T E   S E T Z E N         *
20402 REM ****************************************
20410 IF PX<0 OR PX>319 THEN RETURN
20420 IF PY<0 OR PY>199 THEN RETURN
```

```
20430 HX=8*INT(PX/8)
20440 HY=320*INT(PY/8)+(PY AND 7)
20460 BX=2↑(7-(PX AND 7))
20470 H1=8192+HX+HY
20480 POKEH1,PEEK(H1) OR BX
20490 RETURN
```

Wie wir in Kapitel 4.1.1 schon gesehen haben, ist das Setzen eines Punktes nicht mit einem einfachen Befehl 'POKE X+Y,PU' getan, sondern es muß zunächst das Byte bestimmt werden, indem der Punkt gesetzt werden soll. außerdem noch das entsprechende Bit im Byte. Da die Angaben der X und Y Koordinaten aber im Bereich zwischen 0 und 319 (für X) und 0 und 199 (für Y) liegen, wie wir oben gesehen haben aber keine lineare Durchnumerierung des Farbspeichers erfolgt, muß ein relativ kompliziertes Unterprogramm geschrieben werden.

Um eventuelle Fehler von vornherein auszuschließen (Illegal Quantity Error) wird als erstes geprüft ob für die X und Y Koordinaten der zulässige Bereich nicht überschritten wird. Dann wird eine Hilfsvariable HX mit der Nummer der Spalte besetzt, die im 'Textmodus' den X. Punkt beinhalten würde (vgl Abbildung).

Im weiteren wird in der Variablen HY die Nummer des Bytes im Grafikspeicher abgelegt, die sich wie folgt errechnet: Zunächst wird die Zeile ausgerechnet, wie sie im Textmodus lauten würde, und dieser Wert wird dann mit 320 multipliziert, weil je Zeile 320 Byte zur Verfügung stehen. D.h. eine Zeile im Textmodus umfaßt 40 Zeichen zu je 8 Byte/64 Bits. Faßt man jetzt ein einzelnes Zeichen, wie es im Textmodus vorkommen könnte, zeilenweise auf, so gibt der Ausdruck 'PY AND 7' an, die wievielte der acht Zeilen den Punkt enthält. Dies geschieht indem Y-Werte größer als sieben mit dem AND-Befehl ausgeblendet werden, und nur noch die letzten drei Bits der in ein Byte umgewandelten Zahl zur weiteren Berechnung herangezogen werden.

Als letztes wird noch die Hilfsvariable BX besetzt, die angibt welcher Punkt innerhalb des gewünschten Bytes gesetzt werden muß. Auch hier werden wieder die letzten drei Bits durch den Ausdruck 'AND 7' zur Berechnung herangezogen, und die Funktion '2 hoch 7 - ...' gibt noch das entsprechende Bit im Byte an.

Dann wird noch in der Variablen H1 die absolute Speicher

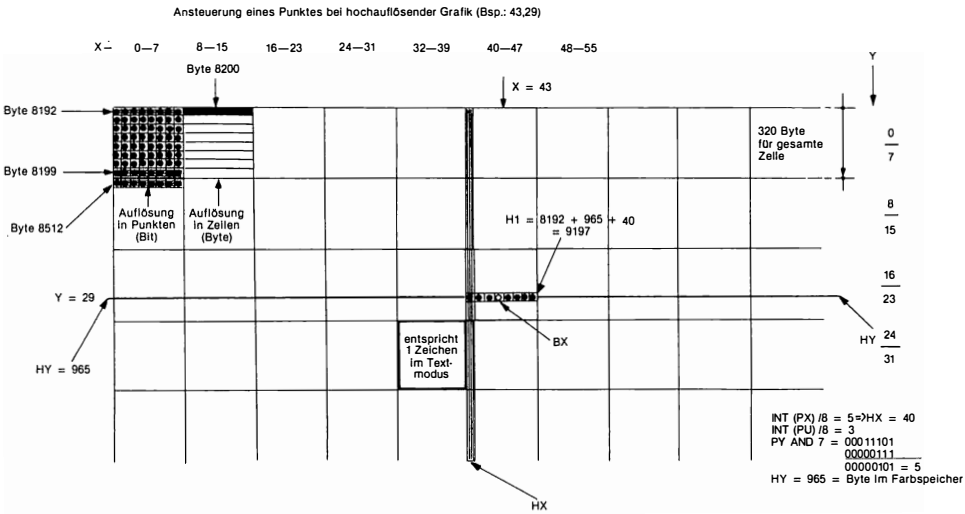

Ansteuerung eines Punktes bei hochauflösender Grafik (Bsp.: 43,29)
X
0—7
8—15
16—23
24—31
32—39
40—47
48—55
Y
Byte 8200
X = 43
Byte 8192
Byte 8199
Byte 8512
Auflösung in Punkten (Bit)
Auflösung in Zeilen (Byte)
320 Byte für gesamte Zelle
0
7
8
15
16
23
24
31
H1 = 8192 + 965 + 40 = 9197
Y = 29
HY = 965
entspricht 1 Zeichen im Text-modus
BX
HY
HX
INT (PX) /8 = 5=>HX = 40
INT (PU) /8 = 3
PY AND 7 = 00011101
00000111
00000101 = 5
HY = 965 = Byte im Farbspeicher

Bild 4.1.2

zelle im RAM angegeben, indem zu dem Grundwert 8192 noch die eben errechneten Werte hinzu addiert werden. Als letztes wird der in BX errechnete Punkt in diesem Byte gesetzt. Dies geschieht indem zunächst das Byte ausgelesen wird, und das gewünschte Bit mittels 'oder' mit dem ausgelesenen Byte verknüpft wird. Das bedeutet, daß egal ob ein Punkt gesetzt ist oder nicht, auf jeden Fall immer zum Schluß dieses Bit gesetzt ist.

Zum Schluß noch ein paar Beispiele zum Aufruf des Unterprogrammes:

```
1100 REM ****************************************
1101 REM *    P U N K T E   S E T Z E N         *
1102 REM ****************************************
1110 PX=1:PY=1:GOSUB20400
1120 PX=10:PY=10:GOSUB20400
1130 PX=100:PY=100:GOSUB20400
1140 PX=100:PY=150:GOSUB20400
1150 PX=100:PY=160:GOSUB20400
1160 PX=100:PY=170:GOSUB20400
1170 PX=110:PY=170:GOSUB20400
1180 PX=120:PY=170:GOSUB20400
1190 PX=130:PY=170:GOSUB20400
```

PUNKT SETZEN **20400 - 20490**

Variablen:

Name	Typ	Bereich	Bedeutung
BX	H	0...255	Hilfsvariable zur Berechnung der Bit-position im Byte
H1	H	8192...16383	Hilfsvariable für Byte im RAM
HX	H	0...319	Hilfsvariable zur Berechnung der X-Position (Byte / Zeile)
HY	H	0...64000	Hilfsvariable zur Berechnung der Y-Position (Byte in Zeile)

PX	E	0...319	Punkt X-Koordinate	
PY	E	0...199	Punkt Y-Koordinate	

Unterprogrammaufrufe :

in	nach	Zweck
keine		

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
20410	RETURN	PX LT 0 oder PX GT 319	
20420	RETURN	PY LT 0 oder PY GT 199	
20490	RETURN	Normales Ende	

4.2 Unterprogramme für den Grafikeinsatz

Übungsziel:

- Grafikausgabe von einfachen geometrischen Figuren
- Unterprogrammaufrufe mit mehreren Hierarchie-Ebenen
- Parameterübergabe an Unterprogramme

In den folgenden Unterkapiteln sollen einige immer wiederkehrende geometrische Formen behandelt werden, die mit entsprechender Parametervorbesetzung sehr einfach von einem Hauptprogramm aufgerufen werden können. Wie Sie sicher feststellen werden, ist diese Ausgabe von Grafikzeichen am Bildschirm sehr langsam, so daß wir an dieser Stelle schon auf Kapitel 6.2 verweisen wollen, wo die selben Routinen noch in Maschinensprache (Assembler) dargestellt sind. Für das Verständnis ist es jedoch, einfacher sich zunächst in Basic sich zu veranschaulichen, was in diesen Unterprogrammen passiert.

4.2.1 Linie ziehen

```
20610 IF EX=AX OR EY=AY THEN H2=1:GOTO20640
20620 IF ABS(EY-AY) < ABS(EX-AX) THEN H2=1: GOTO 20640
20630 H2=ABS(EX-AX)/ABS(EY-AY)
20640 IFEX<AXTHENH2=H2*-1
20650 H3=1
20660 IFEY<AYTHENH3=-1
20670 IFEX=AXTHEN20740
20680 FORLL=AXTOEXSTEPH2
20690 PX=INT(LL+.5)
20700 PY=((LL-AX)*(EY-AY)/(EX-AX))+AY
20710 GOSUB20400
20720 NEXT
20730 RETURN
20740 PX=AX
20750 FORPY=AYTOEYSTEPH3
20760 GOSUB20400
20770 NEXT
20780 RETURN
```

Mit dem vorliegenden Unterprogramm und unter Zuhilfenahme des Unterprogramms zum Setzen von Punkten am Bildschirm, wollen wir es ermöglichen, durch Angabe der Anfangskoordinaten (AX, AY) und der Endkoordinaten (EX, EY) eine Linie am Bildschirm zu zeigen. Wie beim Commodore 64 intern auch, wird die Y-Achse am oberen Rand festgelegt und nicht, wie bei kartesischen Koordinaten üblich, am unteren Rand.

Das Ziehen von Linien ist ein sehr wichtiges Unterprogramm da z.B. das Zeichnen von Rechtecken, Segmenten, Quadern oder Radien in Kreisen bzw. Ellipsen, alle auf dieses Unterprogramm zurückgeführt werden können (vergleiche Bild 4.2.1). Sogar das Ausfüllen von Rechtecken (Block) und Quadern geschieht über das Programm zum Ziehen von Linien.

Das Ziehen von Linien ist durchaus kein triviales Problem, da folgende Punkte besonders berücksichtigt werden müssen:

- Schrittweite; ist die Schrittweite zu groß, erhält man keine voll ausgefüllte Linie, ist die Schrittweite zu klein, dauert das Zeichnen zu lange.
- Negative Steigung; da das Linienziehen am einfachsten in einer FOR...NEXT-Schleife durchzuführen

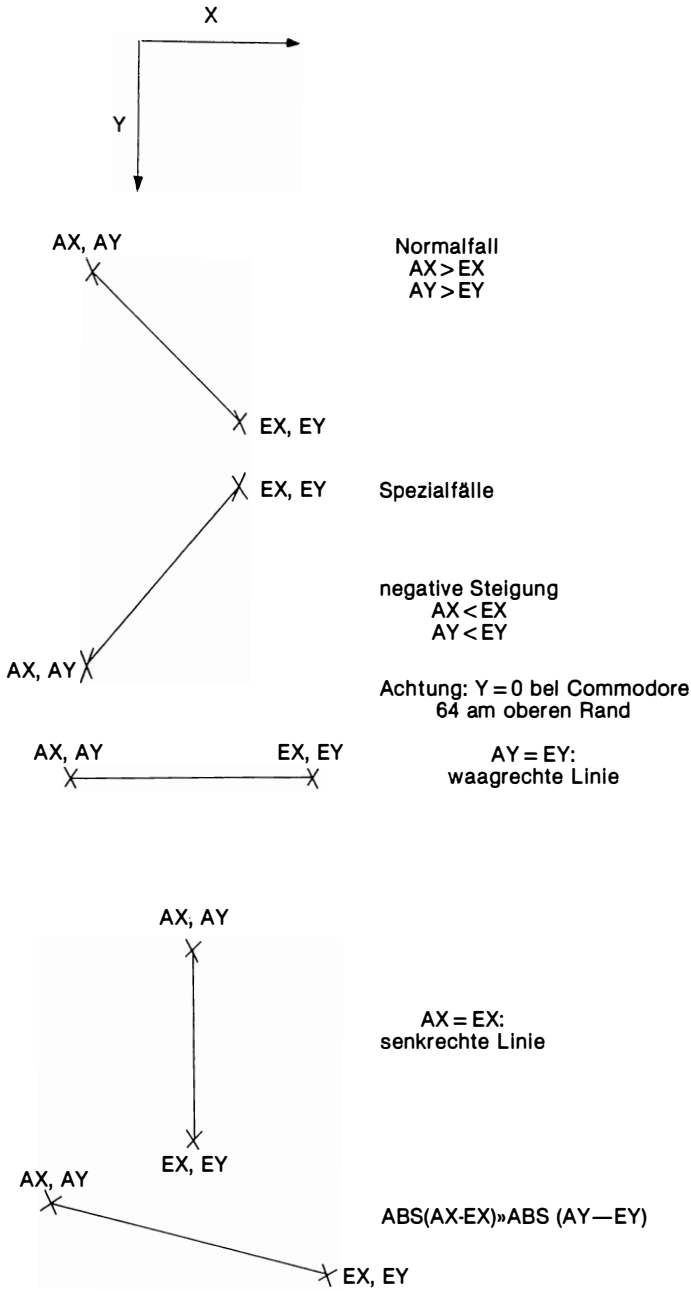


Bild 4.2.1: Prinzipielle Möglichkeiten zum Ziehen von Linien

ist, diese aber in der Regel immer vom kleinsten zum größten Argument gehen, sind negative Steigungen bei Linien gesondert zu behandeln.

Das Ziehen von Linien wollen wir anhand des Verhältnisses von Anfangs- zu Endpunkt und der daraus resultierenden Steigung mit der mathematischen Formel

$$\frac{Y - Y1}{Y2 - Y1} = \frac{X - X1}{X2 - X1}$$

bewirken. Als Eingabeparameter für das Unterprogramm sollen die vier Variablen AX, EX, AY und EY dienen, die jeweils die X und Y - Koordinaten von Anfangs- und Endpunkt enthalten. Zunächst wollen wir uns der Bestimmung der Schrittweite widmen. Ist eine senkrechte (EX=AX) oder waagerechte Linie (EY=AY) zu ziehen, so ist die Schrittweite für die Schleife zum Setzen der einzelnen Punkte gleich '1' zu setzen, was auch das Optimum darstellt. Das weitere Berechnen der Schrittweite kann somit übersprungen werden.

Da wir die Linie in einer FOR...NEXT-Schleife später für alle X-Werte durchlaufen wollen, kann auch die Schrittweite von eins beibehalten werden, wenn die Zahl der Einheiten auf der Y-Achse (ABS(EY-AY)) kleiner ist als die Zahl der Einheiten auf der X-Achse (ABS(EX-AX)). Die Schrittweite '1' ist für eine durchgezogene Linie mit Sicherheit ausreichend, jedoch ist bei großen Unterschieden die Rechenzeit zum Ziehen der Linie größer als erforderlich. Hier ist auch ein Ansatz für Sie für eine Änderung: Man optimiere die Schrittweite. Ist z.B. das Verhältnis der absoluten Beträge in Zeile 20620 größer zwei, so genügt es auch jeden zweiten Punkt auf der Linie nur zu berechnen und die Schrittweite kann somit verdoppelt werden. Man überlege sich ein allgemeines Verfahren zur optimierten Bestimmung der Variablen H2.

Einen Schritt in diese Richtung macht schon die Zeile 20630, in der die Schrittweite optimiert wird wenn das Verhältnis Einheiten auf der X-Achse / Einheiten auf der Y-Achse zu Ungunsten des Abstandes auf der Y-Achse ausfällt. Um zu verhindern, daß eine gestrichelte Linie gezogen wird, muß hier die Schrittweite auf jeden Fall kleiner als '1' angegeben werden.

Ist das Ende der Linie auf der X-Achse vor dem Anfang der Linie gelegen (negative Steigung) so ist die Schrittweite umzukehren (Zeile 20640). Ebenso wenn der Endpunkt auf der Y-Achse 'vor' dem Anfangspunkt liegt.

Wenn die Anfangs-X-Koordinate und End-X-Koordinate gleich sind, so würde die folgende Schleife nur ein einziges Mal durchlaufen, sodaß für diesen Spezialfall eine Schleife für alle Y-Werte durchlaufen werden muß (Zeilen 20740 bis 20770).

In den Zeilen 20680 bis 20720 wird für alle X-Koordinaten die jeweilige Y-Koordinate anhand oben angeführter Gleichung errechnet und ein entsprechender Punkt gesetzt. Die X-Koordinate wird noch entsprechend auf eine ganze Zahl gerundet, da durch die Schrittweite H2 auch gebrochene Zahlen auftreten können.

Mit einem kurzen Programmstück wollen wir das Ziehen von Linien testen, wobei Sie jedoch beachten müssen, daß Sie vorher die Grafik einschalten müssen und wie in Kapitel 4.1 erläutert die entsprechenden RAM-Bereiche besetzen müssen (vergleiche Zusammenhang in Kapitel 4.2.9). Mit dem Programmstück ab Zeile 1200 wird im Prinzip ein Quader am Bildschirm ausgegeben.

```
1200 REM ******************************************
1201 REM *    L I N I E N   Z I E H E N           *
1202 REM ******************************************
1210 AX=40:AY=40:EX=100:EY=40:GOSUB20600
1211 AX=60:AY=25:EX=120:EY=25:GOSUB20600
1212 AX=40:AY=40:EX=60:EY=25:GOSUB20600
1220 AX=100:AY=40:EX=100:EY=70:GOSUB20600
1221 AX=120:AY=25:EX=120:EY=55:GOSUB20600
1222 AX=100:AY=40:EX=120:EY=25:GOSUB20600
1230 AX=100:AY=70:EX=40:EY=70:GOSUB20600
1231 AX=120:AY=55:EX=60:EY=55:GOSUB20600
1232 AX=100:AY=70:EX=120:EY=55:GOSUB20600
1240 AX=40:AY=70:EX=40:EY=40:GOSUB20600
1241 AX=60:AY=55:EX=60:EY=25:GOSUB20600
1242 AX=40:AY=70:EX=60:EY=55:GOSUB20600
```

LINIE	20600 - 20780

Variablen:

Name	Typ	Bereich	Bedeutung
AX	E	0...319	Anfang X-Koordinate

AY	E	0...199	bei Linie Anfang Y-Koordinate bei Linie
EX	E	0...319	Ende X-Koordinate bei Linie
EY	E	0...199	Ende Y-Koordinate bei Linie
H2	H	Dezimalzahl	Hilfsvariable für Schrittweite b. Linien
H3	H	Dezimalzahl	Hilfsvariable für Schrittweite b. Linien
LL	H	AX...EX	Laufvariable bei Linien
PX	P	0...319	Punkt X-Koordinate
PY	P	0...199	Punkt Y-Koordinate

Unterprogrammaufrufe :

in	nach	Zweck
20710	20400	Punkt zeichnen
20760	20400	Punkt zeichnen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
20730	RETURN	EX NE AX	
20780	RETURN	EX = AX	

4.2.2 Rechteck zeichnen

```
20810 REM * OBERE LINIE *
20820 AX=OX
20830 AY=OY
20840 EX=UX
20850 EY=OY
20860 GOSUB20600
20870 REM * RECHTE SEITE *
20880 AX=EX
20890 AY=EY
20900 EX=UX
```

```
20910 EY=UY
20920 GOSUB20600
20930 REM * UNTERE LINIE *
20940 AX=EX
20950 AY=EY
20960 EX=OX
20970 EY=UY
20980 GOSUB20600
20990 REM * LINKE SEITE *
21000 AX=EX
21010 AY=EY
21020 EX=OX
21030 EY=OY
21040 GOSUB20600
21050 RETURN
```

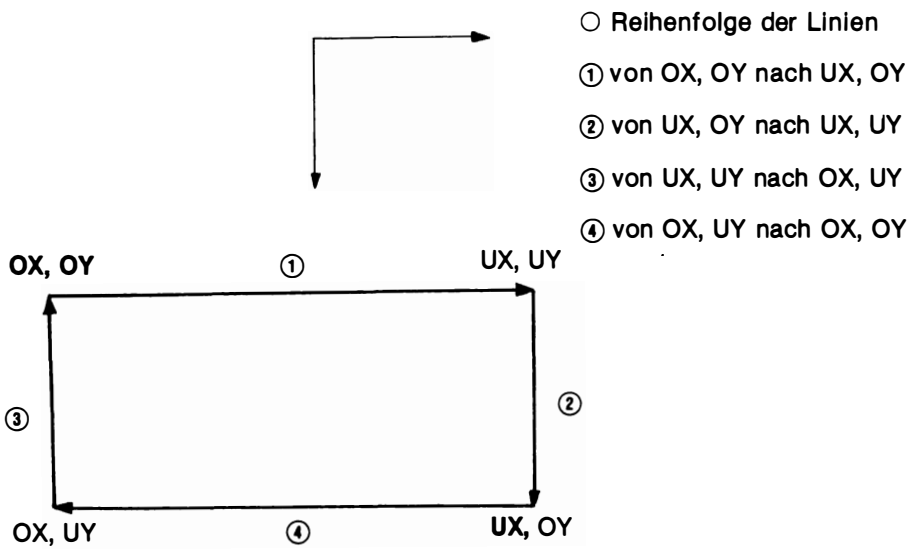


Bild 4.2.2: Veranschaulichung der Variablen und des Ablaufes am Rechteck

Wie wir an dem letzten Beispielprogramm gesehen haben, ist es eine aufwendige Sache einen Quader aus lauter Linien zusammenzusetzen. Zunächst wollen wir jedoch einmal das vordere und hintere Rechteck in ein Unterprogramm bringen. Nachdem das Linien zeichnen schon vorhanden ist, ist das Zeichnen eines Rechteckes keine Schwierigkeit mehr. Durch zwei Koordinatenpunkte läßt sich ein Rechteck eindeutig bestimmen. Diese Punkte müssen natürlich 'linear unabhängig' sein, wie der Fachausdruck bei den Mathematikern dafür lautet. In unserem Falle haben wir den oberen, linken Punkt und den unteren, rechten Punkt eines Rechteckes ausgewählt, womit die beiden anderen Punkte auch festliegen. Die Koordinaten des oberen linken Punktes wollen wir mit OX und OY bezeichnen und die Koordinaten des Punktes unten rechts mit UX und UY.

Wie aus der Abbildung ersichtlich, ergibt sich dann sehr einfach das entsprechende Programmstück in dem nur die Variablen AX, AY, EX und EY vorbesetzt werden und dann das Unterprogramm zum Ziehen einer Linie aufgerufen wird.

Das Besetzen von Variablen nimmt zwar nicht sehr viel Rechenzeit in Anspruch und auch nicht viel Speicherplatz, aber wer will, kann sich überlegen, wie durch vertauschen der Reihenfolge in der die Linien gezeichnet werden, Zuweisungen und somit Programmzeilen, Speicherplatz und Rechenzeit gespart werden kann. Zum Abschluß noch ein kurzes Testprogramm:

```
1300 REM ************************************
1301 REM *    R E C H T E C K   Z E I C H N E N *
1302 REM ************************************
1310 OX=120:OY=100:UX=140:UY=100:GOSUB20800
```

```
!=========================================================!
!                                                         !
!    RECHTECK ZEICHNEN                 20800 - 21050      !
!                                                         !
!=========================================================!
!                                                         !
! Variablen:                                              !
!                                                         !
!---------------------------------------------------------!
! Name ! Typ  ! Bereich          ! Bedeutung              !
!---------------------------------------------------------!
! AX   !  P   ! 0...319          ! Anfang X-Koordinate    !
!      !      !                  ! bei Linie              !
! AY   !  P   ! 0...199          ! Anfang Y-Koordinate    !
!      !      !                  ! bei Linie              !
!=========================================================!
```

EX	P	0...319	Ende **X**-Koordinate bei Linie	
EY	P	0...199	Ende **Y**-Koordinate bei Linie	
OX	E	0...319	**O**bere **X**-Koordinate bei Rechteck (links) und Block	
OY	E	0...199	**O**bere **Y**-Koordinate bei Rechteck (links) und Block	
UX	E	0...319	**U**ntere **X**-Koordinate bei Rechteck (rechts) und Block	
UY	E	0...199	**U**ntere **Y**-Koordinate bei Rechteck (rechts) und Block	

Unterprogrammaufrufe :

in	nach	Zweck
20860	20600	Linie ziehen
20920	20600	Linie ziehen
20980	20600	Linie ziehen
21040	20600	Linie ziehen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
21050	RETURN	Normales Ende	

4.2.3 Quader zeichnen

```
21210 REM * VORDERES RECHTECK *
21220 OX=X1
21230 OY=Y1
21240 UX=X2
21250 UY=Y2
```

```
21260 GOSUB20800
21270 REM * HINTERES RECHTECK *
21280 OX=X3
21290 OY=Y3
21300 UX=X2-X1+X3
21310 UY=Y2-Y1+Y3
21320 GOSUB20800
21330 REM * LINIE OBEN LINKS *
21340 AX=X1
21350 AY=Y1
21360 EX=X3
21370 EY=Y3
21380 GOSUB20600
21390 REM * LINIE OBEN RECHTS *
21400 AX=X2
21410 AY=Y1
21420 EX=X2-X1+X3
21430 EY=Y3
21440 GOSUB20600
21450 REM * LINIE UNTEN RECHTS *
21460 AX=X2
21470 AY=Y2
21480 EX=X2-X1+X3
21490 EY=Y2-Y1+Y3
21500 GOSUB20600
21510 REM * LINIE UNTEN LINKS *
21520 AX=X1
21530 AY=Y2
21540 EX=X3
21550 EY=Y2-Y1+Y3
21560 GOSUB20600
21570 RETURN
```

Nachdem wir in den beiden vorigen Kapiteln das Linienziehen und Rechtecke zeichnen besprochen haben, ist es nun sehr einfach auch einen Quader zu zeichnen, denn ein Quader besteht im Prinzip aus nichts anderem als aus dem vorderen und hinteren Rechteck, sowie vier Verbindungslinien jeweils an den Kanten.

Für die Bestimmung eines Quaders reichen drei geschickt gewählte Punkte aus. Bei der Anzahl von mnemotechnischen Variablen, die wir bisher verwendet haben und auch noch brauchen werden, ist es in diesem Falle das einfachste die drei Punkte mit (X1,Y1),(X2,Y2) und (X3,Y3) zu bezeichnen. Näheres geht aus Abbildung 4.2.3 hervor. Der besseren Übersicht halber, wurden auch noch die Variablen X1, X2, X3, Y1, Y2 und Y3 in ihrer Vektordarstellung beigefügt um die benötigten Berechnungen optisch besser zu gestalten.

Betrachtet man sich die Abbildung so wird die Wirkungs

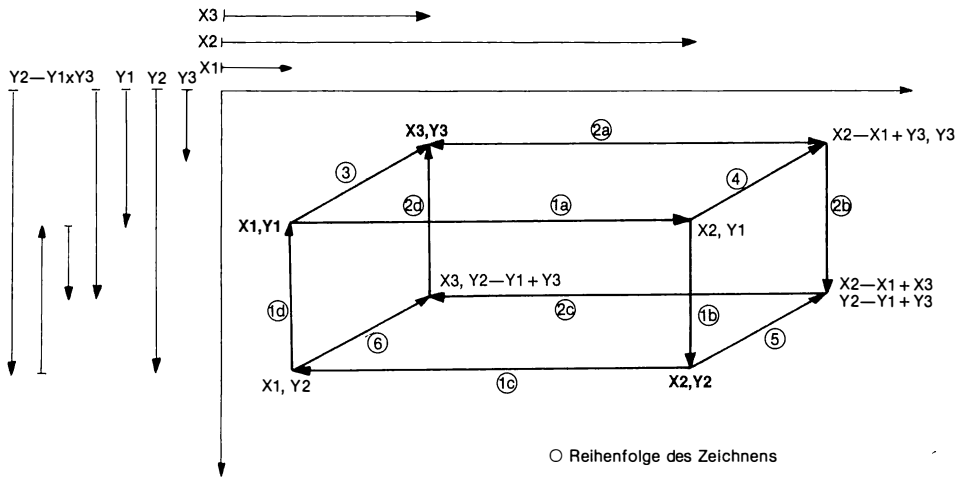


Bild 4.2.3: Veranschaulichung der Gegebenheiten an einem Quader

weise des Programmstückes sofort ersichtlich. Zunächst werden die beiden Rechtecke gezeichnet, wobei hierzu die Aufrufparameter OX, OY, UX und UY mit den entsprechenden Werten besetzt werden und dann das Unterprogramm zum Rechteck zeichnen aufgerufen wird. In den vier weiteren Abschnitten werden, wie in den Bemerkungen angegeben, die vier Linien (in der Zeichnung numeriert von 3 bis 6) durch einen Unterprogrammaufruf gezeichnet, nachdem die entsprechenden Werte für die Linie (AX, AY, EX und EY) vorbesetzt wurden.

An dieser Stelle kann man sich auch überlegen, was in den Programmen geändert werden muß, um Parallelogramme bzw. Quader aus einem Blickwinkel zu zeichnen, wo die Vorderseite keine rechten Winkel enthält. Das Testprogramm:

```
1410 X1=200:Y1=50
1420 X2=230:Y2=70
1430 X3=212:Y3=40
1440 GOSUB21200
```

QUADER ZEICHNEN **21200 - 21570**

Variablen:

Name	Typ	Bereich	Bedeutung
AX	P	0...319	Anfang **X**-Koordinate bei Linie
AY	P	0...199	Anfang **Y**-Koordinate bei Linie
EX	P	0...319	**E**nde **X**-Koordinate bei Linie
EY	P	0...199	**E**nde **Y**-Koordinate bei Linie
OX	P	0...319	**O**bere **X**-Koordinate bei Rechteck (links)
OY	P	0...199	**O**bere **Y**-Koordinate bei Rechteck (links)
UX	P	0...319	**U**ntere **X**-Koordinate bei Rechteck (rechts)
UY	P	0...199	**U**ntere **Y**-Koordinate bei Rechteck (rechts)
X1	E	0...319	**1.** **X**-Koordinate bei Quader (oben links)
X2	E	0...319	**2.** **X**-Koordinate bei Quader (unten rechts)
X3	E	0...319	**3.** **X**-Koordinate bei Quader (oben links hinten)
Y1	E	0...199	**1.** **Y**-Koordinate bei Quader (oben links)
Y2	E	0...199	**2.** **Y**-Koordinate bei Quader (unten rechts)
Y3	E	0...199	**3.** **Y**-Koordinate bei Quader (oben links hinten)

Unterprogrammaufrufe :

in	nach	Zweck
21260	20800	Rechteck zeichnen
21320	20800	Rechteck zeichnen

21380	20600	Linie ziehen
21440	20600	Linie ziehen
21500	20600	Linie ziehen
21560	20600	Linie ziehen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
21570	RETURN	Normales Ende	

4.2.4 Block zeichnen

```
21600 REM ********************************************
21601 REM *    B L O C K          Z E I C H N E N  *
21602 REM ********************************************
21610 AX=OX
21620 EX=UX
21630 FORBL=OYTOUY
21640 AY=BL
21650 EY=BL
21660 GOSUB20600
21670 NEXT
21680 RETURN
```

Mit diesem Unterprogramm wird ein farbig ausgefülltes Rechteck (Block) am Bildschirm ausgegeben. Dies geschieht in dem einzelne Linien parallel am Bildschirm ausgegeben werden. Dazu werden die X-Werte von Beginn und Ende der Linie konstant gehalten. Für die Werte von Y wird eine Schleife mit Schrittweite 1 durchlaufen, so daß ein Block quasi aus ABS(EY-AY) Linien besteht. Die Abbildung zeigt im Prinzip eine Momentaufnahme. Obwohl hier einzelne parallele Linien gezogen werden, kann doch mit den gleichen Variablen wie beim Rechteck gearbeitet werden. Die gleichen Parameter wurden hier deshalb genommen, um nicht noch mehr Variablen für ein umgebendes Hauptprogramm zu sperren. Außerdem ergibt sich dadurch ein Analagon bei dem Aufruf von Block und Rechteck.

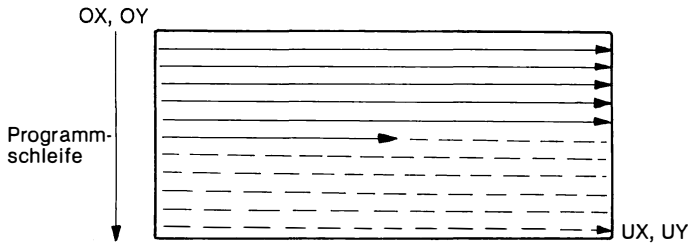


Bild 4.2.4: Vorgehensweise beim Zeichnen eines Blockes

```
1500 REM ****************************************
1501 REM *    B L O C K          Z E I C H N E N *
1502 REM ****************************************
1510 OX=150:OY=80
1520 UX=170:UY=100
1530 GOSUB21600
```

BLOCK ZEICHNEN			21600 - 21680
Variablen:			
Name	Typ	Bereich	Bedeutung
AX	P	0...319	Anfang X-Koordinate bei Linie
AY	P	0...199	Anfang Y-Koordinate bei Linie
BL	H	OY...UY	Laufvariable
EX	P	0...319	Ende X-Koordinate bei Linie
EY	P	0...199	Ende Y-Koordinate bei Linie
OX	E	0...319	Obere X-Koordinate (links)
OY	E	0...199	Obere Y-Koordinate (links)
UX	E	0...319	Untere X-Koordinate

			(rechts)
UY	E	0...199	Untere Y-Koordinate (rechts)

Unterprogrammaufrufe :

in	nach	Zweck
21660	20600	Linie ziehen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
21680	RETURN	Normales Ende	

4.2.5 Kreis und Ellipse zeichnen

```
21810 IFRX>RY THEN H2=RX
21820 IFRY=>RX THEN H2=RY
21830 FORLK=0TO2*πSTEP1/H2
21840 PX=INT(KX+RX*SIN(LK)+.5)
21850 PY=INT(KY+RY*COS(LK)+.5)
21860 GOSUB20400
21870 NEXT
21880 RETURN
```

Kreis und Ellipse können zusammen behandelt werden, da der Kreis eine Sonderform der Ellipse darstellt. Der einzige Unterschied bei einer Ellipse besteht darin, daß zwei Radien, je einer für die X- und die Y-Richtung, gegeben sein müssen. Diese Radien wollen wir mit RX und RY bezeichnen. Bei einem Kreis würde die Bedingung RX=RY erfüllt sein. Ähnlich wie bei der Linie stellt sich auch bei Kreis / Ellipse wieder das Problem der Schrittweite, in welchem Abstand die Punkte errechnet werden sollen. Das in diesem Unterprogramm beschriebene System geht von dem Umfang aus, der dem größeren Radius zugeordnet ist. Beim Zeichnen der Bogen mit dem längeren Radius ergibt sich optimal eine

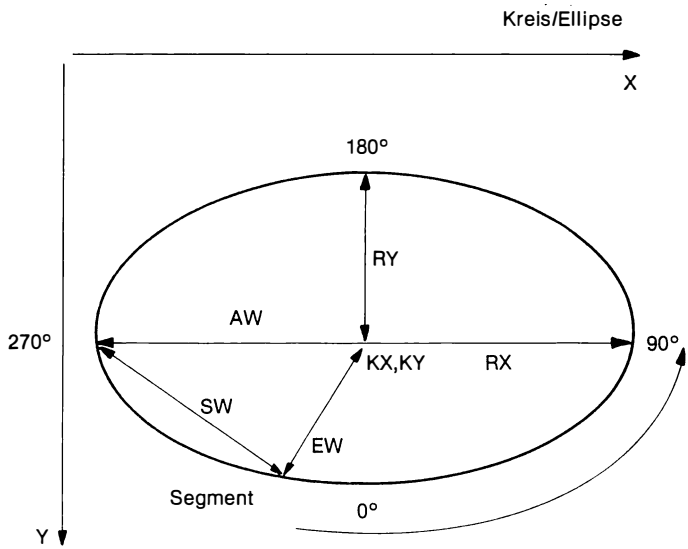


Bild 4.2.5: Zusammenhänge zwischen Kreis/Ellipse, Segment und Radius

ausreichend ausgefüllte Kurve, jedoch bei Kreisbogen die mehr zu dem kürzeren Radius tendieren geht das Zeichnen etwas langsamer, da einzelne Punkte mehrfach berechnet werden. Der geneigte Leser mag sich anhand des Verhältnisses der beiden Radien zueinander eine optimale Schrittweite überlegen. In dem dargestellten Unterprogramm wurde von der Überlegung ausgegangen, daß eine Ellipse mit zwei verschiedenen Radien im Umfang auf jeden Fall kleiner ist, als ein Kreis mit dem größeren der beiden Radien. Da der Umfang eines Kreises mit dem größeren Radius 2*PI*H2 beträgt,die Schleife von Hause aus aber von 0 bis 2*PI läuft, kann als Schrittweite durchaus 1/H2 herangezogen werden.

Wie das Unterprogramm zum Zeichnen einer Linie, so greift auch das Unterprogramm zum Zeichnen eines Kreises auf das Unterprogramm zum Setzen eines Punktes zurück. Der zu setzende Punkt errechnet sich aus den im Programm angegebenen Formeln wobei KX und KY jeweils die Mittelpunktkoordinaten des Kreises sind, und die Winkelfunktionen Sinus für die X-Richtung und Cosinus für die Y-Richtung für ein korrektes Ergebnis jeweils noch mit RX und RY multipliziert werden müssen. Das kurze Testprogramm befindet sich auf der nächsten Seite.

```
1600 REM ****************************************
1601 REM *    K R E I S        Z E I C H N E N *
1602 REM ****************************************
1610 KX=200:KY=150
1620 RX=25:RY=10
1630 GOSUB21800
```

```
!=========================================================!
!                                                         !
!    KREIS ZEICHNEN                  21800 - 21880        !
!                                                         !
!=========================================================!
!                                                         !
! Variablen:                                              !
!                                                         !
!---------------------------------------------------------!
! Name ! Typ  ! Bereich          ! Bedeutung              !
!---------------------------------------------------------!
! H2   !  H   ! Dezimalzahl      ! Hilfsvariable für      !
!      !      !                  ! Schrittweite           !
! KX   !  E   ! 0...319          ! Kreismittelpunkt       !
!      !      !                  ! X-Koordinate bei Kreis!
!      !      !                  ! und Ellipse            !
! KY   !  E   ! 0...199          ! Kreismittelpunkt       !
!      !      !                  ! Y-Koordinate bei Kreis!
!      !      !                  ! und Ellipse            !
! LK   !  H   ! 0...2 * PI       ! Laufvariable bei       !
!      !      !                  ! Kreis und Ellipse      !
! RX   !  E   ! -159...159       ! Radius in X-Richtung   !
! RY   !  E   ! -99...99         ! Radius in Y-Richtung   !
!=========================================================!
!                                                         !
! Unterprogrammaufrufe :                                  !
!                                                         !
!---------------------------------------------------------!
! in    ! nach  ! Zweck                                   !
!---------------------------------------------------------!
! 21860 ! 20400 ! Punkt setzen                            !
!=========================================================!
!                                                         !
! Verzweigungen nach außen :                              !
!                                                         !
!---------------------------------------------------------!
! in Ze ! nach  ! Bedingung         ! Bemerkung           !
!---------------------------------------------------------!
! 21880 ! RETURN! Normales Ende     !                     !
!=========================================================!
```

4.2.6 Segmente zeichnen

```
22220 FUNKTION=AUTOSTART
22230 H2=(W+SW)*2*PI/360
22240 H3=(W+SW)*2*PI/360
22250 AX=INT(KX+RX*SIN(H2)+.5)
22260 AY=INT(KY+RY*COS(H2)+.5)
22270 EX=INT(KX+RX*SIN(H3)+.5)
22280 EY=INT(KY+RY*COS(H3)+.5)
22290 GOSUB20600
22300 NEXT
22310 RETURN
```

Auch das Zeichnen von Segmenten ist wieder auf das Zeichnen von Linien zurückzuführen, jedoch ähnelt die Berechnung des Anfangspunktes und des Endpunktes der Linie den Funktionen, die im letzten Kapitel benötigt wurden. Zur Berechnung eines Kreisausschnittes sind relativ viele Variablen an das Unterprogramm zu übergeben. Dies sind zum einen die für den Kreis benötigten Variablen KX und KY, für den Mittelpunkt und RX und RY für die Radien bzw. der Ellipse, sowie der Anfangswinkel des Ausschnittes und der Endwinkel (AW,EW) und die Schrittweite (SW) in Grad, für welche die Linien gezogen werden sollen .

In den beiden Hilfvariablen H2 und H3 wird der eingegebene Winkel in Grad für jeweils die Anfangs- und Endpunkte eines Segmentes umgerechnet, so daß auch anschließend wieder die Winkelfunktionen Sinus und Kosinus herangezogen werden können. Der Anfangs- und der Endpunkt des Segmentes bilden dann auch wieder die Parameter AX, AY, EX und EY zum Aufrufen des Unterprogramms zum Ziehen eine Linie.

Das Testprogramm:

```
1710 KX=100:KY=150
1720 RX=55:RY=30
1730 AW=0:EW=200
1740 SW=40
1750 GOSUB22200
```

SEGMENTE ZEICHNEN 22200 - 22310

Variablen:

Name	Typ	Bereich	Bedeutung
AX	P	0...319	Anfang X-Koordinate bei Linie
AY	P	0...199	Anfang Y-Koordinate bei Linie
AW	E	0...360	Anfangswinkel
EX	P	0...319	Ende X-Koordinate bei Linie
EY	P	0...199	Ende Y-Koordinate bei Linie
EW	E	0...360	Endwinkel
H2	H	0...2 * PI	Hilfsvariable für Winkelumrechnung
KX	E	0...319	Kreismittelpunkt X-Koordinate bei Kreis Ellipse und Segment
KY	E	0...199	Kreismittelpunkt Y-Koordinate bei Kreis Ellipse und Segment
LS	H	AW...EW	Laufvariable
RX	E	-159...159	Radius in X-Richtung
RY	E	-99...99	Radius in Y-Richtung
SW	E	0...360	Schrittweite

Unterprogrammaufrufe :

in	nach	Zweck
22290	20600	Linie ziehen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
22310	RETURN	Normales Ende	

4.2.7 Quader ausfüllen

```
22410 AX=X1
22420 EX=X2
22430 FORBL=Y1TOY2
22440 AY=BL
22450 EY=BL
22460 GOSUB20600
22470 NEXT
22480 AY=Y1
22490 EY=Y3
22500 FORBL=X1TOX2
22510 AX=BL
22520 EX=BL+X3-X1
22530 GOSUB20600
22540 NEXT
22550 AX=X2
22560 EX=X2+X3-X1
22570 FORBL=Y1TOY2
22580 AY=BL
22590 EY=BL-Y1+Y3
22600 GOSUB20600
22610 NEXT
22620 RETURN
```

Beim Unterprogramm zum Ausfüllen eines Quaders wird zunächst ein Block für die Vorderfront gezeichnet. Das Aufrufen des Unterprogrammes zum Zeichnen eines Blockes wird an dieser Stelle nicht aufgerufen, da dann weitere vier Variablen hätten vorbesetzt werden müssen. Ähnlich wie das Zeichnen eines Blockes geht das Ausfüllen eines Quaders vor sich. Wie gesagt wird zunächst die Vorderfläche gezeichnet, dann die obere Fläche und zm Schluß die rechte Fläche, wobei diese auch jeweils durch parallel geführte Linien farbig ausgefüllt werden.

Bei der Vorderfläche werden auch, wie beim Block, die X-Werte festgehalten und eine Schleife für alle Y-Werte durchlaufen. Anders sieht das bei der oberen Fläche aus; dort werden die Y-Werte festgehalten und eine Schleife für alle X-Werte wird durchlaufen, sodaß die Fläche von links nach rechts ausgefüllt wird. Im dritten Abschnitt erfolgt das Ausfüllen der Fläche wieder von oben nach unten, so daß auch hier wieder die X-Werte konstant sind und eine Schleife für alle Y-Werte durchlaufen wird. Auf der nächsten Seite wieder das Testprogramm.

```
1800 REM ****************************************
1801 REM *    Q U A D E R   A U S F U E L L E N *
1802 REM ****************************************
1810 X1=100:Y1=50
1820 X2=130:Y2=70
1830 X3=112:Y3=40
1840 GOSUB22400
```

QUADER AUSFÜLLEN **22400 - 22620**

Variablen:

Name	Typ	Bereich	Bedeutung
AX	P	0...319	Anfang **X**-Koordinate bei Linie
AY	P	0...199	Anfang **Y**-Koordinate bei Linie
BL	H	Y1...Y2 / X1...X2	Laufvariable
EX	P	0...319	Ende **X**-Koordinate bei Linie
EY	P	0...199	Ende **Y**-Koordinate bei Linie
X1	E/A	0...319	**1.** X-Koordinate bei Quader (oben links)
X2	E/A	0...319	**2.** X-Koordinate bei Quader (unten rechts)
X3	E/A	0...319	**3.** X-Koordinate bei Quader (oben links hinten)
Y1	E/A	0...199	**1.** Y-Koordinate bei Quader (oben links)
Y2	E/A	0...199	**2.** Y-Koordinate bei Quader (unten rechts)
Y3	E/A	0...199	**3.** Y-Koordinate bei Quader (oben links hinten)

Unterprogrammaufrufe :		
in	nach	Zweck
22460	20600	Linie ziehen
22530	20600	Linie ziehen
22600	20600	Linie ziehen

Verzweigungen nach außen :			
in Ze	nach	Bedingung	Bemerkung
22620	RETURN	Normales Ende	

4.2.8 Radius zeichnen

```
22810 AX=KX
22820 AY=KY
22830 H2=RW*2*π/360
22840 EX=INT(KX+RX*SIN(H2)+.5)
22850 EY=INT(KY+RY*COS(H2)+.5)
22860 GOSUB20600
22870 RETURN
```

Auch das Einzeichnen eines Radius in einen Kreis bzw. eine Ellipse ist wieder das Ziehen einer Linie. Da ein Radius definitionsgemäß vom Kreismittelpunkt ausgeht, werden auch die Werte für den Anfang der Linie mit den Mittelpunkts-Koordinaten besetzt. Anschließend erfolgt die Berechnung des Punktes auf dem Kreisumfang analog der Berechnung bei den Segmenten, da ja der Endpunkt eines Radius gleichzeitig auch der Beginn oder das Ende eines Segmentes sein kann.

Testprogramm für alle vorher gezeichneten Segmente:

```
1910 KX=100:KY=150
1920 RX=55:RY=30
1930 RW=0
1931 GOSUB22800
```

```
1940 RW=40
1941 GOSUB22800
1950 RW=80
1951 GOSUB22800
1960 RW=120
1961 GOSUB22800
1970 RW=160
1971 GOSUB22800
```

RADIUS ZEICHNEN **22800 - 22870**

Variablen:

Name	Typ	Bereich	Bedeutung
AX	P	0...319	Anfang **X**-Koordinate bei Linie
AY	P	0...199	Anfang **Y**-Koordinate bei Linie
EX	P	0...319	Ende **X**-Koordinate bei Linie
EY	P	0...199	Ende **Y**-Koordinate bei Linie
H2	H	0... 2 * PI	Hilfsvariable für Winkelumrechnung
KX	E	0...319	Kreismittelpunkt X-Koordinate bei Kreis Ellipse, Segment und Radius
KY	E	0...199	Kreismittelpunkt Y-Koordinate bei Kreis Ellipse, Segment und Radius
RW	E	0...360	Radiuswinkel von 0 Grad an gerechnet
RX	E	-159...159	Radius in **X**-Richtung bei Kreis, Ellipse, Segment und Radius
RY	E	-99...99	Radius in **Y**-Richtung bei Kreis, Ellipse, Segment und Radius

Unterprogrammaufrufe :		
in	nach	Zweck
22860	20600	Linie ziehen

Verzweigungen nach außen :			
in Ze	nach	Bedingung	Bemerkung
22870	RETURN	Normales Ende	

4.2.9 Zusammenfassung

Um das Nachschlagen zu erleichtern und eine besseren Überblick zu geben, haben wir im folgenden das Programmlisting der Kapitel 4.1 und 4.2 vorgestellt und die Tabelle der Variablen nochmals als Ganzes abgedruckt. Da bis auf die Eingabeparameter alle Variablen innerhalb der Unterprogramme tabu für ein umgebendes Hauptprogramm sind, ist eine zusammenfassende Tabelle aller Variablen der Unterprogramme sehr hilfreich bei Verwendung der Zeilen ab 20000 als Unterprogramm.

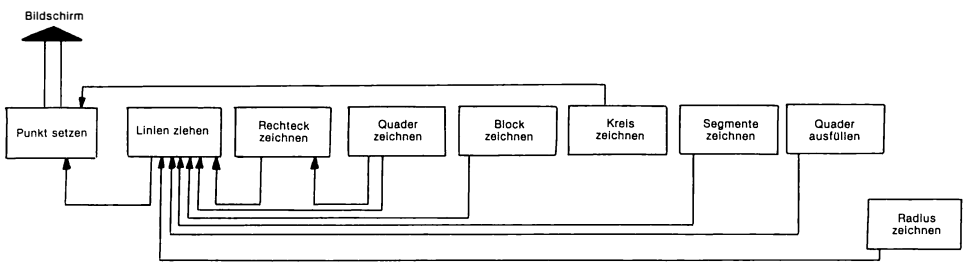


Bild 4.2.9: Zusammenhang der grafischen Unterprogramme

```
10 POKE53280,11
20 POKE53281,11
30 V=53248        : REM ADRESSE VIC
1000 REM ******************************************
1001 REM *                                        *
1002 REM *    B E I S P I E L A U F R U F E       *
1003 REM *                                        *
1004 REM ******************************************
1005 REM :
1007 REM ******************************************
1008 REM *    G R A F I K    E I N                *
1009 REM ******************************************
1010 ZF=7
1020 HF=6
1030 GOSUB20010
1099 GETA$:IFA$=""THEN1099
1100 REM ******************************************
1101 REM *    P U N K T E   S E T Z E N           *
1102 REM ******************************************
1110 PX=1:PY=1:GOSUB20400
1120 PX=10:PY=10:GOSUB20400
1130 PX=100:PY=100:GOSUB20400
1140 PX=100:PY=150:GOSUB20400
1150 PX=100:PY=160:GOSUB20400
1160 PX=100:PY=170:GOSUB20400
1170 PX=110:PY=170:GOSUB20400
1180 PX=120:PY=170:GOSUB20400
1190 PX=130:PY=170:GOSUB20400
1199 GETA$:IFA$=""THEN1199
1200 REM ******************************************
1201 REM *    L I N I E N   Z I E H E N           *
1202 REM ******************************************
1210 AX=40:AY=40:EX=100:EY=40:GOSUB20600
1211 AX=60:AY=25:EX=120:EY=25:GOSUB20600
1212 AX=40:AY=40:EX=60:EY=25:GOSUB20600
1220 AX=100:AY=40:EX=100:EY=70:GOSUB20600
1221 AX=120:AY=25:EX=120:EY=55:GOSUB20600
1222 AX=100:AY=40:EX=120:EY=25:GOSUB20600
1230 AX=100:AY=70:EX=40:EY=70:GOSUB20600
1231 AX=120:AY=55:EX=60:EY=55:GOSUB20600
1232 AX=100:AY=70:EX=120:EY=55:GOSUB20600
1240 AX=40:AY=70:EX=40:EY=40:GOSUB20600
1241 AX=60:AY=55:EX=60:EY=25:GOSUB20600
1242 AX=40:AY=70:EX=60:EY=55:GOSUB20600
1299 GETA$:IFA$=""THEN1299
1300 REM ******************************************
1301 REM *    R E C H T E C K   Z E I C H N E N *
1302 REM ******************************************
1310 OX=120:OY=100:UX=140:UY=130:GOSUB20800
1399 GETA$:IFA$=""THEN1399
1400 REM ******************************************
```

```
1401 REM *    Q U A D E R       Z E I C H N E N *
1402 REM ****************************************
1410 X1=200:Y1=50
1420 X2=230:Y2=70
1430 X3=212:Y3=40
1440 GOSUB21200
1499 GETA$:IFA$=""THEN1499
1500 REM ****************************************
1501 REM *    B L O C K         Z E I C H N E N *
1502 REM ****************************************
1510 OX=150:OY=80
1520 UX=170:UY=100
1530 GOSUB21600
1599 GETA$:IFA$=""THEN1599
1600 REM ****************************************
1601 REM *    K R E I S         Z E I C H N E N *
1602 REM ****************************************
1610 KX=200:KY=150
1620 RX=25:RY=10
1630 GOSUB21800
1699 GETA$:IFA$=""THEN1699
1700 REM ****************************************
1701 REM *    S E G M E N T E   Z E I C H N E N *
1702 REM ****************************************
1710 KX=100:KY=150
1720 RX=55:RY=30
1730 AW=0:EW=200
1740 SW=40
1750 GOSUB22200
1798 GETA$:IFA$=""THEN1799
1799 GOTO1900
1800 REM ****************************************
1801 REM *    Q U A D E R   A U S F U E L L E N *
1802 REM ****************************************
1810 X1=100:Y1=50
1820 X2=130:Y2=70
1830 X3=112:Y3=40
1840 GOSUB22400
1899 GETA$:IFA$=""THEN1899
1900 REM ****************************************
1901 REM *    R A D I U S       Z E I C H N E N *
1902 REM ****************************************
1910 KX=100:KY=150
1920 RX=55:RY=30
1930 RW=0
1931 GOSUB22800
1940 RW=40
1941 GOSUB22800
1950 RW=80
1951 GOSUB22800
1960 RW=120
```

```
1961 GOSUB22800
1970 RW=160
1971 GOSUB22800
1999 GETA$:IFA$=""THEN1999
9000 REM ****************************************
9001 REM *    G R A F I K    A U S               *
9002 REM ****************************************
9990 GOSUB20200
9999 END
20000 REM ****************************************
20001 REM *                                      *
20002 REM * G R A F I S C H E   R O U T I N E N  *
20003 REM *                                      *
20004 REM ****************************************
20005 REM :
20006 REM ****************************************
20007 REM *    G R A F I K    E I N               *
20008 REM ****************************************
20020 POKEV+17,59    :REM BYTE: 00111011
20030 POKEV+24,24
20040 FORLV=1024TO2023
20050 POKELV,HF+16*ZF
20060 NEXT
20070 FORLV=8192TO16383
20080 POKELV,0
20090 NEXT
20100 POKE53280,HF
20110 RETURN
20200 REM ****************************************
20201 REM *    G R A F I K    A U S               *
20202 REM ****************************************
20210 POKEV+17,27    :REM BYTE: 00011011
20220 POKEV+24,21    :REM ????????????????
20230 RETURN
20400 REM ****************************************
20401 REM *    P U N K T E   S E T Z E N         *
20402 REM ****************************************
20410 IF PX<0 OR PX>319 THEN RETURN
20420 IF PY<0 OR PY>199 THEN RETURN
20430 HX=8*INT(PX/8)
20440 HY=320*INT(PY/8)+(PY AND 7)
20460 BX=2↑(7-(PX AND 7))
20470 H1=8192+HX+HY
20480 POKEH1,PEEK(H1) OR BX
20490 RETURN
20600 REM ****************************************
20601 REM *    L I N I E N   Z I E H E N         *
20602 REM ****************************************
20610 IF EX=AX OR EY=AY THEN H2=1:GOTO20640
20620 IF ABS(EY-AY) < ABS(EX-AX) THEN H2=1: GOTO 20640
20630 H2=ABS(EX-AX)/ABS(EY-AY)
```

```
20640 IFEX<AXTHENH2=H2*-1
20650 H3=1
20660 IFEY<AYTHENH3=-1
20670 IFEX=AXTHEN20740
20680 FORLL=AXTOEXSTEPH2
20690 PX=INT(LL+.5)
20700 PY=((LL-AX)*(EY-AY)/(EX-AX))+AY
20710 GOSUB20400
20720 NEXT
20730 RETURN
20740 PX=AX
20750 FORPY=AYTOEYSTEPH3
20760 GOSUB20400
20770 NEXT
20780 RETURN
20800 REM ************************************************
20801 REM *     R E C H T E C K   Z E I C H N E N *
20802 REM ************************************************
20810 REM * OBERE LINIE *
20820 AX=OX
20830 AY=OY
20840 EX=UX
20850 EY=OY
20860 GOSUB20600
20870 REM * RECHTE SEITE *
20880 AX=EX
20890 AY=EY
20900 EX=UX
20910 EY=UY
20920 GOSUB20600
20930 REM * UNTERE LINIE *
20940 AX=EX
20950 AY=EY
20960 EX=OX
20970 EY=UY
20980 GOSUB20600
20990 REM * LINKE SEITE *
21000 AX=EX
21010 AY=EY
21020 EX=OX
21030 EY=OY
21040 GOSUB20600
21050 RETURN
21200 REM ************************************************
21201 REM *     Q U A D E R       Z E I C H N E N *
21202 REM ************************************************
21210 REM * VORDERES RECHTECK *
21220 OX=X1
21230 OY=Y1
21240 UX=X2
21250 UY=Y2
```

```
21260 GOSUB20800
21270 REM * HINTERES RECHTECK *
21280 OX=X3
21290 OY=Y3
21300 UX=X2-X1+X3
21310 UY=Y2-Y1+Y3
21320 GOSUB20800
21330 REM * LINIE OBEN LINKS *
21340 AX=X1
21350 AY=Y1
21360 EX=X3
21370 EY=Y3
21380 GOSUB20600
21390 REM * LINIE OBEN RECHTS *
21400 AX=X2
21410 AY=Y1
21420 EX=X2-X1+X3
21430 EY=Y3
21440 GOSUB20600
21450 REM * LINIE UNTEN RECHTS *
21460 AX=X2
21470 AY=Y2
21480 EX=X2-X1+X3
21490 EY=Y2-Y1+Y3
21500 GOSUB20600
21510 REM * LINIE UNTEN LINKS *
21520 AX=X1
21530 AY=Y2
21540 EX=X3
21550 EY=Y2-Y1+Y3
21560 GOSUB20600
21570 RETURN
21600 REM ******************************************
21601 REM *    B L O C K         Z E I C H N E N *
21602 REM ******************************************
21610 AX=OX
21620 EX=UX
21630 FORBL=OYTOUY
21640 AY=BL
21650 EY=BL
21660 GOSUB20600
21670 NEXT
21680 RETURN
21800 REM ******************************************
21801 REM *    K R E I S         Z E I C H N E N *
21802 REM ******************************************
21810 IFRX>RY THEN H2=RX
21820 IFRY=>RX THEN H2=RY
21830 FORLK=0TO2*πSTEP1/H2
21840 PX=INT(KX+RX*SIN(LK)+.5)
21850 PY=INT(KY+RY*COS(LK)+.5)
```

```
21860 GOSUB20400
21870 NEXT
21880 RETURN
22200 REM ****************************************
22201 REM *    S E G M E N T E   Z E I C H N E N *
22202 REM ****************************************
22220 FORLS=AWTOEWSTEPSW
22230 H2=LS*2*π/360
22240 H3=(LS+SW)*2*π/360
22250 AX=INT(KX+RX*SIN(H2)+.5)
22260 AY=INT(KY+RY*COS(H2)+.5)
22270 EX=INT(KX+RX*SIN(H3)+.5)
22280 EY=INT(KY+RY*COS(H3)+.5)
22290 GOSUB20600
22300 NEXT
22310 RETURN
22400 REM ****************************************
22401 REM *    Q U A D E R   A U S F U E L L E N *
22402 REM ****************************************
22410 AX=X1
22420 EX=X2
22430 FORBL=Y1TOY2
22440 AY=BL
22450 EY=BL
22460 GOSUB20600
22470 NEXT
22480 AY=Y1
22490 EY=Y3
22500 FORBL=X1TOX2
22510 AX=BL
22520 EX=BL+X3-X1
22530 GOSUB20600
22540 NEXT
22550 AX=X2
22560 EX=X2+X3-X1
22570 FORBL=Y1TOY2
22580 AY=BL
22590 EY=BL-Y1+Y3
22600 GOSUB20600
22610 NEXT
22620 RETURN
22800 REM ****************************************
22801 REM *    R A D I U S     Z E I C H N E N *
22802 REM ****************************************
22810 AX=KX
22820 AY=KY
22830 H2=RW*2*π/360
22840 EX=INT(KX+RX*SIN(H2)+.5)
22850 EY=INT(KY+RY*COS(H2)+.5)
22860 GOSUB20600
22870 RETURN
```

GRAF **1000 - 22070**

Variablen:

Name	Typ	Bereich	Bedeutung
A$	H	1 Zeichen	für Warteschleife
AX	E/P	0...319	Anfang X-Koordinate bei Linie
AY	E/P	0...199	Anfang Y-Koordinate bei Linie
AW	E/P	0...360	Anfangswinkel bei Segmenten
BL	H	0Y...UY	Laufvariable bei Blocks und ausgefüllten Quadern
BX	H	0...255	Hilfsvariable zur Berechnung der Bitposition im Byte
EX	E/P	0...319	Ende X-Koordinate bei Linie
EY	E/P	0...199	Ende Y-Koordinate bei Linie
EW	E/P	0...360	Endwinkel bei Segmenten
HF	G	0...15	Hintergrundfarbe
H1	H	8192...16383	Hilfsvariable für Byte im RAM beim Punkt-setzen
H2	H	Dezimalzahl	Hilfsvariable für Schrittweite b. Linien
H3	H	Dezimalzahl	Hilfsvariable für Schrittweite b. Linien
HX	H	0...319	Hilfsvariable zur Berechnung der X-Position (Byte / Zeile)
HY	H	0...12800	Hilfsvariable zur Berechnung der Y-Position (Byte in Zeile)
KX	E/P	0...319	Kreismittelpunkt X-Koordinate bei Kreis Ellipse, Segment und Radius
KY	E/P	0...319	Kreismittelpunkt Y-Koordinate bei Kreis

				Ellipse, Segment und Radius
LK	H	0...2 * PI		Laufvariable bei Kreis und Ellipse
LL	H	AX...EX		Laufvariable bei Linien
LS	H	AW...EW		Laufvariable bei Segmenten
OX	E/P	0...319		Obere X-Koordinate bei Rechteck (links) und Block
OY	E/P	0...199		Obere Y-Koordinate bei Rechteck (links) und Block
PX	E/P	0...319		Punkt X-Koordinate
PY	E/P	0...199		Punkt Y-Koordinate
RW	E/P	0...360		Radiuswinkel bei Radius
RX	E/P	-159...159		Radius in X-Richtung bei Kreis, Ellipse, Segment und Radius
RY	E/P	-99...99		Radius in Y-Richtung bei Kreis, Ellipse, Segment und Radius
SW	E/P	0...360		Schrittwinkel bei Segmenten
UX	E/P	0...319		Untere X-Koordinate bei Rechteck (rechts) und Block
UY	E/P	0...199		Untere Y-Koordinate bei Rechteck (rechts) und Block
V	G	konstant 53248		Adresse VIC
X1	E/P	0...319		1. X-Koordinate bei Quader (oben links)
X2	E/P	0...319		2. X-Koordinate bei Quader (unten rechts)
X3	E/P	0...319		3. X-Koordinate bei Quader (oben links hinten)
Y1	E/P	0...199		1. Y-Koordinate bei Quader (oben links)
Y2	E/P	0...199		2. Y-Koordinate bei Quader (unten rechts)
Y3	E/P	0...199		3. Y-Koordinate bei Quader (oben links hinten)
ZF	G	0...15		Zeichenfarbe (Punkte)

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung

Unterprogrammaufrufe :

in	nach	Zweck
1030	20010	Hochauflösende Grafik einschalten
1110	20400	Punkt ausgeben
1120	20400	Punkt ausgeben
1130	20400	Punkt ausgeben
1140	20400	Punkt ausgeben
1150	20400	Punkt ausgeben
1160	20400	Punkt ausgeben
1170	20400	Punkt ausgeben
1180	20400	Punkt ausgeben
1190	20400	Punkt ausgeben
1210	20600	Linie ziehen (für Quader)
1211	20600	Linie ziehen (für Quader)
1212	20600	Linie ziehen (für Quader)
1220	20600	Linie ziehen (für Quader)
1221	20600	Linie ziehen (für Quader)
1222	20600	Linie ziehen (für Quader)
1230	20600	Linie ziehen (für Quader)
1231	20600	Linie ziehen (für Quader)
1232	20600	Linie ziehen (für Quader)
1240	20600	Linie ziehen (für Quader)
1241	20600	Linie ziehen (für Quader)
1242	20600	Linie ziehen (für Quader)
1310	20800	Rechteck zeichnen
1440	21200	Quader zeichnen (Rahmen)
1530	21600	Block zeichnen
1630	21800	Kreis zeichnen
1750	22200	Segmente zeichnen
1840	22400	Quader zeichnen (ganz)
1931	22800	Radius zeichnen
1941	22800	Radius zeichnen
1951	22800	Radius zeichnen
1961	22800	Radius zeichnen
1971	22800	Radius zeichnen
9990	20200	Grafik ausschalten

20710	20400	Punkt zeichnen	
20760	20400	Punkt zeichnen	
20860	20600	Linie ziehen	
20920	20600	Linie ziehen	
20980	20600	Linie ziehen	
21040	20600	Linie ziehen	
21260	20800	Rechteck zeichnen	
21320	20800	Rechteck zeichnen	
21380	20600	Linie ziehen	
21440	20600	Linie ziehen	
21500	20600	Linie ziehen	
21560	20600	Linie ziehen	
21660	20600	Linie ziehen	
21860	20400	Punkt setzen	
22290	20600	Linie ziehen	
22460	20600	Linie ziehen	
22530	20600	Linie ziehen	
22600	20600	Linie ziehen	
22860	20600	Linie ziehen	

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
keine			

4.3 Tortengrafik

Übungsziel:

- Verwendung der in Kapitel 4.2 beschriebenen Unterprogramme
- Normierung von Eingabewerten
- Grafische Gestaltung

```
1010 ZF=7
1020 MF=6
2000 INPUT"■ANZAHL DER FELDER";AF
2010 DIMWF(AF)
2020 DIMGR(AF)
2030 FORI=1TOAF
2040 PRINT"WERT FELD ";I,
```

```
2050 INPUTWF(I)
2060 SU=SU+WF(I)
2070 NEXT
2080 PRINTSU
2090 FORI=1TOAF
2100 GR(I)=INT(WF(I)*360/SU+.5)
2110 NEXT
2120 GOSUB20000
2130 KX=200     : REM EVTL MIT INPUT EINLESEN
2140 KY=100     : REM EVTL MIT INPUT EINLESEN
2150 RX=45      : REM EVTL MIT INPUT EINLESEN
2160 RY=20      : REM EVTL MIT INPUT EINLESEN
2170 : REM EVTL ELLIPSE FUER ANDERE DARSTELLUNG
2180 GOSUB21800
2190 RW=0
2200 SU=0
2210 GOSUB22800
2220 SW=1
2230 FORI=1TOAF-1
2240 SU=SU+GR(I)
2250 RW=SU
2260 GOSUB12200
2270 NEXT
2280 GETA$:IFA$=""THEN2280
2290 GOSUB20200:STOP
9990 GOSUB20200
9999 END
```

Um eventuellen Fehlern gleich vorzubeugen: Die Unterprogramme ab Zeile 20000 der Kapitel 4.2 sind Bestandteil dieses Programms, auch wenn sie nicht extra wieder mit ausgedruckt werden (aus Platzgründen).

Mit diesem Programm werden beliebig hohe Eingabewerte normiert und als Tortengrafik dargestellt. Zunächst wird die Anzahl der Felder in die Variable AF eingelesen, womit gleichzeitig auch die Anzahl der einzugebenden Werte festliegt. Dann werden zwei Felder dimensioniert: WF() für die Eingabewerte der Felder selbst und GR() für die normierten Werte. Im folgenden werden die Daten eingelesen und in der Variablen SU aufsummiert. Die Normierung geschieht mit dem üblichen Dreisatz, indem für die Summe aller eingegebenen Werte ein Vollkreis von 360 Grad angesetzt wird.

Dann wird in Zeile 2120 die Grafik eingeschaltet und die Werte für den Kreis/die Ellipse werden vorbesetzt. Bei den angegebenen Daten handelt es sich um eine Ellipse in der rechten Bildschirmhälfte. Soll die Tortengrafik mittels eines Kreises dargestellt werden, so sind RX und RY mit

gleicher Höhe zu wählen. Daraufhin wird ein Vollkreis am Bildschirm ausgegeben, mittels des Unterprogramms ab Zeile 11400. Im Folgenden dient die Variable SU zum aufsummieren der bereits von der Tortengrafik belegten Gradzahl am Kreis. Zu bemerken sei hier noch, daß 0 Grad bei dem Commodore 64 unten am Kreis sind. In den Zeilen 2190 bis 2210 wird der erste Radius für 0 Grad gedruckt. Zeile 2220 kann weggelassen werden, sie dient nur zum Besetzen der Schrittweite, falls die Tortengrafik segmentweise gezeichnet werden soll.

Anschließend wird für alle Eingabewerte bis auf den Letzten jeweils ein Radius entsprechend der normierten Gradzahl gezogen. Die Größe des letzten Feldes ergibt sich automatisch aus der Differenz zwischen 0 Grad = 360 Grad und dem vorhergehenden Radius. Den Abschluß bildet in Zeile 2280 noch eine Warteschleife bis eine Taste gedrückt wird, anschließend wird die Grafik wieder ausgeschaltet und das Programm angehalten.

TORTENGRAFIK **2000 - 2290**

Variablen:

Name	Typ	Bereich	Bedeutung
A$	H	1 Zeichen	Warteschleife
AF	G	Integer	Anzahl der Stücke
I	H	1...AF	Laufvariable
KX	P	0...319	X-Koordinate Kreis-mittelpunkt (Ellipse)
KY	P	0...199	Y-Koordinate Kreis-mittelpunkt (Ellipse)
RX	P	0...319	Radius in X-Richtung
RY	P	0...199	Radius in Y-Richtung
RW	P	0...360	Radiuswinkel
SU	G	Dezimalzahl	Summe der eingegebenen Werte
SW	P	0...360	Schrittweite bei Segmenten

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
GR()	1...AF	G	Dezimalzahlen	normierter Betrag in Grad
WF()	1...AF	G	Dezimalzahlen	Eingabewerte

Unterprogrammaufrufe :

in	nach	Zweck
2120	20010	Grafikteil - Grafik einschalten
2180	21800	Grafikteil - Kreis ausgeben
2210	22800	Grafikteil - Radius ausgeben (0 Grad)
2260	22800	Grafikteil - Radius ausgeben (Felder)
2290	20200	Grafikteil - Grafik ausschalten

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
2290	STOP	Normales Ende	(wegen 3000-Balk.)

4.4 Balkendiagramme

Übungsziel:

- Darstellung von Balken mittels Block, Rechteck, Quader und ausgefüllten Quadern
- Verwendung der in Kapitel 4.2 vorgestellten Unterprogramme
- Normierung von Eingabewerten

Unter Verzicht auf den Multi-Color-Modus können in dem Programm Balkengrafiken nur einfarbig dargestellt werden - im Gegensatz zu den in Kapitel 2.3 beschriebenen Balkendiagrammen. Jedoch können mit diesem Programm nicht nur Balken, sondern auch Rechtecke, Quader oder ausgefüllte

```
1000 REM *** GRAFIK EIN ***
1010 ZF=7
1020 HF=6
3000 INPUT"ANZAHL DER BALKEN";AB
3010 DIMWB(AB)
3020 DIMHO(AB)
3030 FORI=1TOAB
3040 PRINT"WERT BALKEN ";I,
3050 INPUTWB(I)
3060 SU=SU+WB(I)
3070 IFWB(I)>MATHENMA=WB(I)
3080 NEXT
3090 PRINT"SUMME                    ",SU
3100 INPUT"ABSTAND                  ";A
3110 INPUT"BREITE                   ";BR
3120 INPUT"GRUNDZEILE               ";GZ
3130 INPUT"MAXIMALE HOEHE (ZEILE) ";MH
3140 INPUT"ERHOEHUNG                ";EH
3150 INPUT"BEGINN IN SPALTE         ";BS
3160 PRINT"      1 - BLOCK          "
3170 PRINT"      2 - RECHTECK       "
3180 PRINT"      3 - QUADER         "
3190 PRINT"      4 - QUADER VOLL    "
3200 INPUT" 1 / 2 / 3 / 4           ";AR
3210 IFAR<1ORAR>4THEN3200
3220 IFAR>2THENINPUT"TIEFE                      ";TF
3230 FORI=1TOAB
3240 PR(I)=INT(WB(I)*100/SU+.5)
3250 HO(I)=INT(WB(I)*(GZ-MH-TF-EH*(AB-1))/MA+.5)
3260 NEXT
3270 GOSUB20000
3280 UX=BS-A
3290 FORI=1TOAB
3300 OX=UX+A
3310 UX=OX+BR
3320 OY=GZ-HO(I)
3330 UY=GZ-EH*(I-1)
3340 GOSUB3700
3350 NEXT
3360 GOTO 9990
3700 IFAR=1THENGOSUB21600:RETURN
3710 IFAR=2THENGOSUB20800:RETURN
3720 IFAR=3THENGOSUB3740:GOSUB21200:RETURN
3730 IFAR=4THENGOSUB3740:GOSUB22400:RETURN
3740 X1=OX
3750 Y1=OY
3760 X2=UX
3770 Y2=UY
3780 X3=OX+TF
3790 Y3=OY-TF
3800 RETURN
```

Quader dargestellt werden.

Zunächst wird die Anzahl der Balken erfragt, und darauf werden zwei Felder - zur Speicherung der Eingabewerte und der auf die maximale Höhe normierten Werte - dimensioniert. Dann werden die Werte eingelesen und wie in Kapitel 4.3 auch eine Summe gebildet. Gleichzeitig wird auch noch das maximale Element der Eingabewerte in Zeile 3070 herausgefunden.

In den Zeilen 3090 bis 3220 erfolgt die Erfassung der für die Diagramme benötigten Werte. Dies sind offensichtlich einige mehr als in Kapitel 4.3, jedoch werden auch hier mehr Möglichkeiten zugelassen. Die zu berücksichtigenden Werte sind:

- A - Abstand der Balken voneinander; der Abstand der Balken untereinander kann auch negativ sein, so daß sich eine geschachtelte Abbildung ergibt
- B - Breite der Balken in Bildschirmpunkten
- GZ - Grundzeile; die Grundzeile ist die unterste Punktreihe des ersten Balkens bzw. aller Balken, wenn die Erhöhung 0 ist
- MH - maximale Höhe; die maximale Höhe gibt die oberste beschreibbare Zeile (Zeile hier im Sinne von Punktzeile) im Gegensatz zu der Grundzeile an. Die Grafik erscheint also maximal zwischen Grundzeile und maximaler Höhe.
- EH - Erhöhung, die Erhöhung gibt an, um wieviele Punkte jeder weitere Balken sich von der Grundzeile mehr entfernen soll, als der vorhergehende Punkt. Im Zusammenhang mit der Variablen A (Abstand) läßt sich so eine 'Fluchtlinie' erzeugen.
- BS - Beginn Spalte, um Grafiken nicht immer am linken Bildschirmrand beginnen zu lassen, sondern damit auch mehrere Grafiken auf dem Bildschirm nacheinander erzeugt werden können, kann man noch die äußerst linke Punktzeile der Grafik in Variablen BS angeben.
- AR - Art der Darstellung, in dieser Variablen wird Ziffernmäßig die Art der Darstellung verschlüsselt dargestellt (siehe Listing).
- TF - Tiefe, die Variable TF wird nur besetzt, wenn ein Quader bzw. gefüllter Quader gezeichnet werden soll. TF gibt dann an, um wieviele Punkte jeweils in X und Y Richtung der Quader nach hinten gezeichnet werden soll (optisch hinten ist am Bildschirm oben).

In Zeile 3210 befindet sich noch eine Plausibilitätsprüfung für die Art der Darstellung. Ab Zeile 3230 befindet sich die Normierung der Zahlen, wobei wie folgt vorgegangen wird: Der größte Eingabewert erhält (im Gegensatz zu der Berechnung bei der Tortengrafik) die maximale Höhe zugewiesen. Alle anderen werden **zu diesem Balken relativ** dargestellt. Die Berechnung der maximalen Höhe eines Balkens sieht zwar etwas kompliziert aus, ist im Prinzip aber ganz einfach. Zunächst kann der Balken die maximale Höhe GZ-MH haben. Von dieser Höhe muß noch die eventuelle Tiefe abgezogen werden, damit bei Quadern nicht über den vorgegebenen Rahmen hinaus gezeichnet wird. Da der größte Balken auch der letzte sein kann, muß - falls eine Erhöhung vorliegt - diese so oft, wie sie gebraucht wird, auch noch von der maximalen Balkenhöhe abgezogen werden.

In Zeile 3260 wird die Grafik eingeschaltet und in Zeile 3280 bis 3340 eine Schleife zum Errechnen der jeweiligen Eckpunkte der Balken und deren Ausgabe durchlaufen. Da der obere Eckpunkt eines Balkens aufgrund des Abstandes und des vorgehenden unteren Eckpunktes berechnet wird, muß noch vorher (in Zeile 3270) der imaginäre untere Eckpunkt vor dem ersten Balken festgelegt werden. Weil der Abstand von der Beginnspalte nicht eingehalten werden soll, ist er zunächst abzuziehen, da er in Zeile 3290 wieder aufaddiert wird.

Wie schon erwähnt, wird die obere (linke) X-Koordinate aus der unteren (rechte) X-Koordinate des letzten Balkens, vermehrt um den Abstand, bestimmt. Die untere X-Koordinate ergibt sich dann aus der oberen X-Koordinate, vermehrt um die gewünschte Breite des Balkens. Die obere Y-Koordinate ist die Grundzeile abzüglich der Höhe des Balkens und einer eventuell vorliegenden Erhöhung. Die untere Y-Koordinate ist der gleiche Ausdruck ohne die Höhe des Balkens.

Im Unterprogramm ab Zeile 3700 wird noch überprüft, welche Darstellungsart gewünscht wird und entsprechend verzweigt, wobei für die Darstellungsarten '3' und '4' jeweils noch die entsprechenden Aufrufparameter X1, X2, X3, Y1, Y2 und Y3 entsprechend besetzt werden.

BALKEN HIRES **3000 - 3800**

Variablen:

Name	Typ	Bereich	Bedeutung
A	G	00...319	Horizontaler Abstand zwischen den Balken
A$	H	1 Zeichen	Warteschleife
AB	G	Ganzzahlig	Anzahl der Balken
AR	G	1...4	Art der Darstellung: 1 - Block 2 - Rechteck 3 - Quader 4 - Quader voll
BR	G	0...319	Breite der Balken
BS	G	0...318	Beginn in Spalte (Linke Seite des ersten Balkens)
EH	G	0...199	Vertikaler Abstand der Balken zwischen den Grundseiten
GZ	G	0...199	Grundzeile des ersten Balkens (Unterste Z.)
I	H	1...AB	Laufvariable
MA	G	Dezimalzahl	Maximum der eingegebenen Werte
MH	G	0...199	Oberste Zeile
SU	G	Dezimalzahl	Summe der eingegebenen Werte
TF	G	0...198	Tiefe bei Quadern nach oben und rechts in gleicher Punktzahl
OX	P	0...319	X-Koordinate oben links
OY	P	0...199	Y-Koordinate oben links
UX	P	0...319	X-Koordinate unten rechts
UY	P	0...199	Y-Koordinate unten rechts
X1	P	0...319	X-Koord. oben links
X2	P	0...319	X-Koord. unten rechts
X3	P	0...319	X-Koord. hinten
Y1	P	0...199	Y-Koord. oben links

Y2	P	0...199	Y-Koord. unten rechts
Y3	P	0...199	Y-Koord. hinten

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
HO()	1...AB	G	Dezimalzahlen	normierte Höhe
WB()	1...AB	G	Dezimalzahlen	Eingabewerte

Unterprogrammaufrufe :

in	nach	Zweck
3270	20010	Grafikteil - Grafik einschalten
3340	3700	intern - Darstellungsart feststellen
3700	21600	Grafikteil - Block ausgeben
3710	20800	Grafikteil - Rechteck ausgeben
3720	3740	intern - Parameter Quader besetzen
3720	21200	Grafikteil - Quader (Linien) ausgeben
3730	3740	intern - Parameter Quader besetzen
3730	22400	Grafikteil - Quader (voll) ausgeben
9995	20200	Grafikteil - Grafik ausschalten

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
9999	END	Normales Ende	

5

Assembler

5. Assembler

Wie wir in Kapitel 4 gesehen haben, kann Basic sehr langsam sein. Sicher haben Sie schon Programme gesehen, die komplexe Bildschirmausgaben in Blitzesschnelle bewältigen. Dazu ist jedoch eine Programmierung auf Maschinenebene erforderlich, im Prinzip eine Bit-Fummelei mit lauter '0' und '1'. Über Assembler alleine kann man schon Bücher schreiben, weil das Gebiet so komplex ist. Dies haben wir auch mit dem Band 4 dieser Buchreihe getan. An dieser Stelle sei deshalb nur eine 'abgemagerte' Version beschrieben, und das Programm wird nicht bis ins Kleinste erläutert. Ein Assembler ist jedoch unabdingbare Voraussetzung für die optimale Nutzung aller Möglichkeiten des Commodore 64.

Ein Assembler ist im Prinzip ein Mittelding zwischen Maschinencode und Basic. Wie Interessierte vielleicht schon wissen, werden von einem Byte aus acht Bits jeweils vier Bit zu einer sogenannten Hexadezimalzahl (Basis 16) zusammengefaßt, die dann eine Bezeichnung entsprechend von 0 bis F erhalten. Dies ist der erste Schritt zur Vereinfachung, da statt vier Bit immer nur noch eine hexadezimale Ziffer geschrieben werden muß. Der nächste Schritt ist eine mnemotechnische Bezeichnung für die zu setzenden Bits z.B. 'LDX' für 'lade in X-Register' oder 'STX' speichere (STORE) in X-Register. Genauso wie bei Basic gibt es auch zusammengesetzte Befehle, wo der Befehlsteil in unserem Beispiel 'LDX' mit diversen Parametern versehen sein kann. Dies sind in der Hauptsache verschiedene Adressierungsarten. Wie schon erwähnt sei hier auf ergänzende Literatur zum näheren Verständnis hingewiesen.

Ziel dieses Buches ist es, lediglich einen Assembler vorzustellen, dessen Verständnis im einzelnen nicht erforderlich ist, lediglich seine Handhabung, um z.B. die in Kapitel 4.2 vorgestellten Unterprogramme schneller zu machen.

Der hier vorgestellte Assembler macht nichts anderes, als diese mnemotechnischen Bezeichnungen (z.B. LDX) in ihren hexadezimalen Code umzuwandeln. Dies sieht am Anfang etwas einfacher aus als es ist, jedoch sind auch die verschiedenen Möglichkeiten der Parameter für die einzelnen Befehle zu beachten. Ein weiterer Schritt zur leichteren Handhabung von Maschinenprogrammen, ist die Möglichkeit der Verwendung von Symbolen, so daß Sie nicht immer die abso-

Speicheradressen angeben müssen, sondern - wie in höheren Programmiersprachen auch - sich ein Symbol für eine Speicherstelle definieren.

Sicher wundert es Sie etwas, daß der Assembler in ein vom Computer sofort ausführbaren Maschinencode in Basic geschrieben ist, aber wie schon gesagt ist es ein Umsetzungs**programm**, und die hierfür einfachste Sprache ist wohl Basic. Eine lohnende Sache wäre es sicherlich, das im Folgenden aufgezeigte Basic-Programm in Assembler zu schreiben, um auch dem - recht langsamen - Assembler auf die Sprünge zu helfen. Fertige Teilprogramme könnten dann bei der Erstellung des Assemblers schon das Assemblieren unterstützen. Das wäre dann so, als wenn Sie mit einem halbfertigen Auto die restlichen Teile einkaufen fahren, geht aber entschieden besser.

Das Kapitel über den Assembler haben wir in vier Bereiche untergliedert. Zunächst gehen wir auf die DATA-Statements ein, die die verschiedenen mnemotechnischen Bezeichnungen wie Sie bei dem 6510/6502 üblich sind inklusive ihrer Parameter angeben.

Als nächstes wird auf häufig verwendete Unterprogramme eingegangen, die aus Rechenzeitgründen am Anfang des Programms stehen.

Hier noch einiges zur näheren Erklärung: Bei einem GOSUB-Befehl sucht der Computer bei einer höheren Zeilennummer als die Aufrufzeile hat, ab der Aufrufzeile und bei einer kleineren Zeilennummer vom Programmanfang. Häufig verwendete Unterprogramme sollten deshalb am Programmanfang stehen, um dem Computer lange Suchzeiten zu ersparen. Dies sind im einzelnen Millisekunden; bei einer entsprechenden Häufigkeit von Aufrufen summiert sich das jedoch schon zu Sekunden, wenn nicht sogar Minuten. Im dritten Teil wird auf die anderen Unterprogramme eingegangen und im vierten Unterkapitel wird letztendlich das Hauptprogramm beschrieben.

5.1 DATA-Statements

Die DATA-Statements liegen im Basic-Programm ab Zeile 54000 bis hin zu Zeile 59200. Die DATA-Statements selbst sind wieder wie folgt untergliedert:

54000 Anzahl der Adressenmodes je Befehlstyp
55000 Angabe der mnemotechnischen Befehle mit ihrem

dreibuchstabigen Code sowie die Nummer des Typs und alle aufgrund des Typs möglichen späteren Maschinencodes
56000 Direktiven; Anweisungen im Assemblerprogramm, die keinen Maschinencode erzeugen
57000 Basic-Schlüsselwörter
58000 Fehlermeldungen

An dieser Stelle seien noch zwei Begriffe erläutert: Source-Code / Quellprogramm ist die Bezeichnung für den im Assembler eingegebenen Text und Objekt-Code bzw. Objekt-Programm ist das später fertige Maschinenprogramm was aus dem Source-Code resultiert.

5.1.1 Adressenmodes

```
54010 DATA1:REM TYP 0 IMPLIED
54020 DATA2:REM TYP 1 (JUMPS) ABSOLUTE,INDIRECT
54030 DATA10:REM #,Z,ZX,ZY,A,AX,AY,IX,IY,AC  TYP 2
54040 DATA0:REM BYTE
54050 DATA0:REM WORD
54060 DATA1:REM TYP 5 RELATIVE BRANCHES
```

Die Adressenmodes werden später in dem Feld AC(0..5) gespeichert. Folgende Anzahlen für Adressenmodes sind möglich:

Typ 0 - 1, d.h. ein dreibuchstabiger Assemblerbefehl des Typ 0, kann nur in einen einzigen Maschinenbefehl umgewandelt werden.

Typ 1 - 2, hauptsächlich Jump-Befehle sind vom Typ 1. Diese können absolut und indirekt adressiert werden

Typ 2 - bis 10, in Typ 2 sind alle möglichen Operationen zusammengefaßt, die in bis zu zehn verschiedene Maschinencodes umgewandelt werden können.

Typ 3 - 0, erzeugt keinen Code, sondern ist eine Direktive zum Reservieren einer Konstanten von einem Byte

Typ 4 - 0, erzeugt ebenfalls keinen Code, sondern dient zum Reservieren einer Konstanten von

zwei Byte (= 1 Word)

Typ 5 - 1, faßt alle Befehle für relative Sprünge zusammen.

5.1.2 Mnemotechnische Befehle

```
55040 DATAADC,2,69,65,75,,6D,7D,79,61,71,
55050 DATAAND,2,29,25,35,,2D,3D,39,21,31,
55060 DATAASL,2,,06,16,,0E,1E,,,,0A
55070 DATABCC,5,90
55080 DATABCS,5,B0
55090 DATABEQ,5,F0
55100 DATABIT,2,,24,,,2C,,,,,
55110 DATABMI,5,30
55120 DATABNE,5,D0
55130 DATABPL,5,10
55140 DATABRK,0,00
55150 DATABVC,5,50
55160 DATABVS,5,70
55170 DATABYT,3
55180 DATACLC,0,18
55190 DATACLD,0,D8
55200 DATACLI,0,58
55210 DATACLV,0,B8
55220 DATACMP,2,C9,C5,D5,,CD,DD,D9,C1,D1,
55230 DATACPX,2,E0,E4,,,EC,,,,,
55240 DATACPY,2,C0,C4,,,CC,,,,,
55250 DATADEC,2,,C6,D6,,CE,DE,,,,
55260 DATADEX,0,CA
55270 DATADEY,0,88
55280 DATAEOR,2,49,45,55,,4D,5D,59,41,51,
55290 DATAINC,2,,E6,F6,,EE,FE,,,,
55300 DATAINX,0,E8
55310 DATAINY,0,C8
55320 DATAJMP,1,4C,6C
55330 DATAJSR,1,20,
55340 DATALDA,2,A9,A5,B5,,AD,BD,B9,A1,B1,
55350 DATALDX,2,A2,A6,,B6,AE,,BE,,,
55360 DATALDY,2,A0,A4,B4,,AC,BC,,,,
55370 DATALSR,2,,46,56,,4E,5E,,,,4A
55380 DATANOP,0,EA
55390 DATAORA,2,09,05,15,,0D,1D,19,01,11,
55400 DATAPHA,0,48
55410 DATAPHP,0,08
55420 DATAPLA,0,68
55430 DATAPLP,0,28
55440 DATAROL,2,,26,36,,2E,3E,,,,2A
```

```
55450 DATAROR,2,,66,76,,6E,7E,,,,6A
55460 DATARTI,0,40
55470 DATARTS,0,60
55480 DATASBC,2,E9,E5,F5,,ED,FD,F9,E1,F1,
55490 DATASEC,0,38
55500 DATASED,0,F8
55510 DATASEI,0,78
55520 DATASTA,2,,85,95,,8D,9D,99,81,91,
55530 DATASTX,2,,86,,96,8E,,,,,
55540 DATASTY,2,,84,94,,8C,,,,,
55550 DATATAX,0,AA
55560 DATATAY,0,A8
55570 DATATYA,0,98
55580 DATATSX,0,BA
55590 DATATXA,0,8A
55600 DATATXS,0,9A
55610 DATAWOR,4
```

In diesen DATA-Statements sind alle mnemotechnischen Befehle des 6510/6502 zusammengefaßt. Jede DATA-Zeile besteht aus dem dreibuchstabigen Befehl, dem Typ, wie er in Kapitel 5.1.1 beschrieben wurde, und den verschiedenen hexadezimalen Werten, die das Maschinenprogramm aufgrund des Codes und den verschiedenen Adressierungen annehmen kann.

Folgende Befehle sind aufgeführt:

ADC - A**D**diere zum Akkumulator mit **C**arry-Flag
AND - UND-Verknüpfung einer Speicherzelle mit dem Akkumulator
ASL - Shift Left; alle Bits um eins nach links schieben, von rechts wird eine Null nachgezogen, und das äußerst linke Bit befindet sich anschließend im Carry-Flag
BCC - **B**ranch if **C**arry **C**lear
BCS - **B**ranch if **C**arry **S**et; verzweige, wenn das Carry-Flag gesetzt ist
BEQ - **B**ranch if **E**Qual; verzweige, wenn Zero-Flag = 1
BIT - Teste Speicherzelle mit Akkumulator
BMI - **B**ranch if **MI**nus; verzweige, wenn Negativ-Flag=1
BNE - **B**ranch if **N**ot **E**qual; verzweige, wenn Zero-Flag = 0
BPL - **B**ranch if **PL**us; verzweige, wenn Negativ-Flag =0
BRK - B**R**ea**K**; Abbruch (Software-Interrupt)
BVC - **B**ranch if O**V**erflow **C**lear; verzweige, wenn kein Überlauf vorhanden ist (Overflow-Flag = 0)
BVS - **B**ranch O**V**erflow **S**et; verzweige, wenn ein Über-

lauf vorhanden ist (Overflow-Flag = 1)
BYT - Reservieren eines **Byt**es (kein Maschinenbefehl)
CLC - **CL**ear **C**arry; Carryflag löschen
CLD - **CL**ear **D**ecimal Flag; Umschalten auf Binärarithmetik
CLI - **CL**ear **I**nterupt Flag; Interrupt-Flag löschen (Interrupt ermöglichen)
CLV - **CL**aer O**V**erflowflag; Überlauf-Flag löschen
CMP - **CoMP**are Accu ; vergleiche Akku mit Speicher
CPX - **ComP**are **X** ; vergleiche X-Register mit Speicher
CPY - **ComP**are **Y** ; vergleiche Y-Register mit Speicher
DEC - **DEC**rement Memory ; Speicherinhalt um eins erniedrigen
DEX - **DE**crement **X** ; X-Register um eins erniedrigen
DEY - **DE**crement **Y** ; Y-Register um eins erniedrigen
EOR - **E**xklusiv-**O**de**r**-Verknüpfung Speicherzelle mit Akkumulator
INC - **INC**rement Memory ; Speicherinhalt un eins erhöhen
INX - **IN**crement **X** ; X-Register um eins erhöhen
INY - **IN**crement **Y** ; Y-Register um eins erhöhen
JMP - **JuMP** ; Unbedingter Sprung
JSR - **J**ump **Sub**routine ; Unterprogramm-Aufruf
LDA - **LoaD** Accu ; Lade Akkumulator mit Speicher
LDX - **LoaD** X ; Lade X-Register mit Speicher
LDY - **LoaD** Y ; Lade Y-Register mit Speicher
LSR - Shift Right;alle Bits um '1' nach rechts schieben, von links wird eine Null nachgezogen, und das äußerst rechte Bit befindet sich anschließend im Carry-Flag
NOP - **N**o **OP**eration ; Leerbefehl
ORA - Oder-Verknüpfung Speicherzelle mit **A**kkumulator
PHA - **Push** **A**ccu ; Akku in Kellerspeicher (Stack)
PHP - **Push** **P**rocessorstatus ; Prozessor-Status-Register in Kellerspeicher
PLA - **Pull** **A**ccu ; Akku aus Kellerspeicher holen
PLP - **Pull** **P**rocessorstatus ; Prozessor-Status-Register aus Kellerspeicher holen
ROL - **RO**tate **L**eft - Rotiere ein Bit linksherum (Akku oder Speicher); Bit 7 geht in das Carry-Flag, das alte Carry-Flag geht nach Bit 0
ROR - **RO**tate **R**ight - Rotiere ein Bit rechtsherum (Akku oder Speicher); Bit 0 geht in das Carry-Flag, das alte Carry-Flag geht nach Bit 7
RTI - **R**e**T**urn from **I**nterrupt - Rückkehr vom Interrupt; restauriert Programmzähler und Prozessorstatus
RTS - **R**e**T**urn from **S**ubroutine - Rückkehr vom Unterprogramm; restauriert nur Programmzähler
SBC - **Sub**trahiere vom Akkumulator mit **C**arry-Flag
SEC - **SE**t **C**arry; Carryflag setzen

SED - **SE**t **D**ecimal Flag; Umschalten auf Dezimalarithmetik
SEI - **SE**t **I**nterupt Flag; Interrupt-Flag setzen (Interrupt verhindern)
STA - **ST**ore **A**ccu; speichere Akkumulator
STX - **ST**ore **X**; speichere X-Register
STY - **ST**ore **Y**; speichere Y-Register
TAX - **T**ransfer **A**ccu into **X**; Akkumulator in X-Register übertragen
TAY - **T**ransfer **A**ccu into **Y**; Akkumulator in Y-Register übertragen
TSX - **T**ransfer **S**tack-Pointer into **X**; Stackpointer in X-Register übertragen
TXA - **T**ransfer **X** into **A**ccu; Akkumulator in X-Register übertragen
TXS - **T**ransfer **X** into **S**tackpointer; X-Register in Stackpointer übertragen
TYA - **T**ransfer **Y** into **A**ccu; Akkumulator in Y-Register übertragen
WOR - Reservieren eines **Word** (Kein Maschinenbefehl)

5.1.3 Direktiven

```
56030 DATAORG,END,INO
```

In der vorliegenden Assemblerversion sind nur drei Direktiven vorhanden, diese sind:

%ORG - erstes absolutes Byte für das Assembler-Programm festlegen
%END - Ende des Sources-Codes
%INO - Startadresse für das Assembler-Programm am Bildschirm erfragen

In den DATA-Statements werden die Direktiven ohne '%' angegeben.

5.1.4 Basic-Keywords

```
57030 DATAEND,FOR,NEXT,DATA,INPUT#,INPUT,DIM
57040 DATAREAD,LET,GOTO,RUN,IF,RESTORE
57050 DATAGOSUB,RETURN,REM,STOP,ON,WAIT,LOAD
57060 DATASAVE,VERIFY,DEF,POKE,PRINT#
57070 DATAPRINT,CONT,LIST,CLR,CMD,SYS,OPEN
```

```
57020 DATA"SYNTAX : FALSCHES MNEMONIC"
57030 DATA"SYNTAX : FALSCHE DIREKTIVE"
57040 DATA"SYNTAX : OPERAND FEHLT"
57050 DATA"SYNTAX : KLAMMERN FALSCH GESETZT"
57060 DATA"SYNTAX : MODEZEICHEN NICHT AN ERSTER STELLE"
57070 DATA"ADRESSIERUNGSART HIER NICHT MOEGLICH"
57080 DATA"KEIN OPERAND ERLAUBT"
57090 DATA"OPERAND MUSS EINE KONSTANTE SEIN"
57100 DATA"OPERAND MUSS EINE MARKE (LABEL) SEIN"
57110 DATA"OPERAND FALSCH SPEZIFIZIERT"
57120 DATA"MARKE ODER KONSTANTE SCHON DEFINIERT"
57130 DATA"AUSDRUCK DARF NUR 2 ARGUMENTE ENTHALTEN"
57140 DATA"WERT ZU GROSS"
57150 DATA"FALSCHE LAENGE EINER HEX- ODER BINAER-ZAHL"
57160 DATA"UNGUELTIGES ZEICHEN IN BINAERZAHL"
57170 DATA"DIESER WERT MUSS BEREITS HIER DEFINIERT SEIN"
57180 DATA"SPRUNG ZU WEIT"
57190 DATA"ZU VIELE FEHLER"
57200 DATA"STARTADRESSE NICHT DEFINIERT"
57210 DATA"STARTADRESSE BEREITS DEFINIERT"
```

Wie interessierte Leser vielleicht schon wissen, werden die Basic-Keywords jeweils auch in einem Byte dargestellt, wie ein einzelnes Zeichen. Diese Codes liegen etwas abseits von den normalen Buchstaben im ASCII-Bereich von 128 bis 203. Deshalb wird auch in dem Feld BK$() dieser Bereich mit den Basic-Keywords besetzt. Auf die Möglichkeiten der einzelnen Basic-Keywords wollen wir hier nicht näher eingehen. Notwendig ist die Verarbeitung der Basic-Keywords, da in den Assembler-Programmen eventuell Basic-Keywords vorkommen können, die da nichts zu suchen haben. Würden Sie zum Beispiel eine Variable 'UNSINN' definieren, so würde diese Variable als 'UN'+'SIN(US)'+'N' gespeichert werden. Um diese Sachen zu korrigieren, müssen die Basic-Keywords später im Programm jeweils überprüft werden, so daß der Text eines Basic-Keywords in den Source-Code wieder anhand dessen Verschlüsselung eingesetzt werden kann.

5.1.5 Fehlermeldungen

In diesem Kapitel werden die Fehlermeldungen beschrieben, die vom Assembler ausgegeben werden, diese sind:

(1) - Syntax-error / kein nmemotechnischer Code
(2) - Syntax-error / keine Direktive
(3) - Syntax-error / Operand fehlt
(4) - Syntax-error / Klammer falsch gesetzt
(5) - Syntax-error / Operand falsch spezifiziert
(6) - angegebene Adressierungsart bei diesem Befehl nicht möglich
(7) - kein Operand erlaubt
(8) - Operand muß eine Konstante sein
(9) - Operand muß eine Marke sein
(10) - Operand hat falsche Form
(11) - Marke bzw. Konstante schon definiert
(12) - Ausdruck enthält mehr als zwei Argumente
(13) - Wert zu groß
(14) - falsche Länge für hexadezimalen bzw. binären Ausdruck
(15) - ungültiges Zeichen in binärem Ausdruck
(16) - dieser Wert muß bereits hier definiert sein
(17) - Sprung zu groß
(18) - zu viele Fehler
(19) - Startadresse nicht definiert
(20) - Startadresse wurde schon definiert

Wer selbst den Assembler ergänzen will, kann auch diese Fehlerliste nach eigenem Ermessen noch ausbauen.

Wer will, kann sich sogar seine mnemotechnischen Bezeichnungen, abweichend von den hier angegebenen Üblichen, selbst wählen, vielleicht mit deutschen Kürzeln. Zwingende Voraussetzung ist jedoch, daß die mnemotechnischen Bezeichnungen drei Buchstaben beinhalten.

Direktiven können von jedem selbst eingebaut werden, ohne daß diese auch dreibuchstabig sein müssen (vgl auch Band 4).

5.2 Häufig benötigte Unterprogramme

Im folgenden werden die Unterprogramme erklärt, die aus Rechenzeitgründen an den Anfang des Programms genommen wurden.

```
 10   Ein Zeichen lesen aus File #2 in I$ und II
 50   Position (A) von A$ in AA$ bestimmen
100   Nachfolgende Blanks von T$ eliminieren
150   Führende Blanks von T$ eliminieren
200   2-stellige Hexzahl (H$)  aus  H bilden
250   4-stellige Hexzahl (HH$) aus HH bilden
350   Wert von 2-stelliger Hexzahl (H$)  nach H
400   Wert von 4-stelliger Hexzahl (HH$) nach HH
500   Name TN$ in Tabelle suchen und Werte bestimmen
600   Check auf mnemotechnischen Code
```

Zeichen aus Datei einlesen

```
15 GET#2,I$
20 IS=ST
25 IFI$=""THENI$=CHR$(0)
30 II=ASC(I$)
35 RETURN
```

Dieses kurze Unterprogramm liest ein Zeichen aus einer Datei (#2) ein und speichert in der Variablen IS den Status des IEC-Bus ab, in I$ das Zeichen selbst und II dessen ASCII-Code. Da von der Floppy eingelesen wird, muß das Zeichen CHR$(0) gesondert behandelt werden, da es von der Floppy als sogenannter Null-String eingelesen wird.

Indexfunktion

```
55 A=0
60 AA=LEN(A$)
65 IFAA>LEN(AA$)THEN85
70 FORA=1TOLEN(AA$)-AA+1
75 IFMID$(AA$,A,AA)<>A$THENNEXT
80 IFA>LEN(AA$)-AA+1THENA=0
85 RETURN
```

Dieses Unterprogramm dient dazu, die Position einer Teil-

zeichenreihe in einer größeren Zeichenreihe zu bestimmen. Dazu wird von Beginn an die größere Teilzeichenreihe stückweise mit der kleineren Zeichenreihe verglichen und das Unterprogramm abgebrochen, sobald Gleichheit vorliegt. In Zeile 65 befindet sich noch eine Plausibiltätsprüfung, da die Indexfunktion natürlich nicht ausgeführt werden kann, wenn die zu suchende Zeichenreihe größer ist als die Zeichenreihe, in der gesucht werden soll.

Die Variable A (Ausgabeparameter) wird auf Null gesetzt, wenn die gesuchte Zeichenreihe nicht gefunden wurde.

Nachfolgende Blanks eliminieren

```
110 IFRIGHT$(T$,1)=" "THENT$=LEFT$(T$,LEN(T$)-1):GOTO110
120 RETURN
```

Dieses kurze Unterprogramm schneidet von hinten von einer gewünschten Zeichenreihe, die in der Variablen T$ übergeben werden muß, in einer Schleife jeweils anhängende Blanks ab. Diese Schleife wird abgebrochen, sobald ein anderes Zeichen als ein Leerzeichen abgefragt wird.

Führende Blanks eliminieren

```
160 IFLEFT$(T$,1)=" "THENT$=MID$(T$,2):GOTO160
170 RETURN
```

Analog zu dem letzten Unterprogramm werden hier von vorne die Blanks abgeschnitten.

Zweistellige Hexadezimalzahl bilden

```
210 H$=MID$(HE$,H/16+1,1)+MID$(HE$,(HAND15)+1,1)
220 RETURN
```

In diesem kurzen Unterprogramm werden aus einer Integerzahl bis 255 zwei hexadezimale Ziffern gebildet.

Vierstellige Hexadezimalzahl bilden

```
260 H=INT(HH/256)
270 GOSUB200
280 HH$=H$
290 H=HH-256*H
300 GOSUB200
310 HH$=HH$+H$
320 RETURN
```

Dieses Unterprogramm funktioniert genauso wie das vorherige jedoch werden hier aus Integerzahlen bis zu Höhe von 32767 vier hexadezimale Ziffern gebildet. Dazu wird das vorherige Programm zweimal für die Werte INT(HH/256) und den Rest dieses Wertes aufgerufen.

Integerzahl aus zweistelliger Hexadezimalzahl

```
360 H=16*FNV(ASC(H$))+FNV(ASC(MID$(H$,2)))
370 RETURN
```

Dieses Programm bildet die Umkehrung zu dem weiter oben beschriebenen Programm und errechnet aus einer zweistelligen Hexadezimalzahl einen Integerwert.

Vierstellige Hexadezimalzahl in Integerwert umwandeln

```
410 H$=LEFT$(HH$,2)
420 GOSUB350
430 HH=256*H
440 H$=RIGHT$(HH$,2)
450 GOSUB350
460 HH=HH+H
470 RETURN
```

Dieses Programm bildet die Umkehrung zu dem Unterprogramm ab Zeile 250, indem eine vierstellige Hexadezimalzahl in einen Integerwert bis 32767 umgewandelt wird.

Tabelle durchsuchen und Wert bestimmen

```
510 TF=0
520 TN$=LEFT$(TN$,8)
530 IFTA=0THENRETURN
540 FORTF=1TOTA
550 IFTN$(TF)<>TN$THENNEXT
560 IFTF>TATHENTF=0:RETURN
570 TV=TV(TF)
```

```
580 TY$=TY$(TF)
590 RETURN
```

In diesem Unterprogramm wird in der Symboltabelle (die vom Benutzer selber definiert wird) ein Wert gesucht. Die Variable TF ist Null, falls nicht gefunden und die Nummer des Symbols in dem Symbolfeld, falls es gefunden wurde. Außerdem werden die Variablen TV, die den Wert des Symbols angibt, und TY$, die den Typ des Symboles angibt, vorbesetzt. Typ eines Symboles kann sowohl Label (Marke) als auch Konstante sein.

Test auf Vorhandensein der mnemotechnischen Bezeichnung

```
610 FORK=1TOK9
620 IFA$<>K$(K)THENNEXT
630 IFK>K9THENK=0
640 KM=KM(K)
650 RETURN
```

In diesem Unterprogramm wird überprüft, ob die im Source-Code eingegebene mnemotechnische Bezeichnung überhaupt zulässig ist, wobei der Variablen K die Nummer des Symboles in dem Feld zugeordnet wird. Ebenfalls wird noch der Typ von dem Code (siehe Kapitel 5.1.1) an die Variable KM übergeben.

5.3 Seltener verwendete Unterprogramme

An dieser Stelle sollen die größeren Unterprogramme beschrieben, wie sie später beim Hauptprogramm benutzt werden. Zum Verständnis des Hauptprogramms sind diese Unterprogramme von entscheidender Bedeutung. Bitte haben Sie Verständnis dafür, daß wir nicht aufs aller genaueste auf die Unterprogramme eingehen können, da sonst der Rahmen dieses Buches gesprengt würde.

5.3.1 Eine Zeile assemblieren

```
10010 C$=""
10020 U$=" "
10030 ER=0
10040 GOSUB150
10050 A$=";"
10060 AA$=T$
10070 GOSUB50
10080 IFATHENT$=LEFT$(T$,A-1)
```

```
10090 GOSUB100
10100 IFLEFT$(T$,1)="%"THENGOSUB13000:RETURN
10110 AA$=T$
10120 A$="="
10130 GOSUB50
10140 IFA=0THEN10290
10150 T0$=LEFT$(T$,A-1)
10160 T$=MID$(T$,A+1)
10170 GOSUB150
10180 SI=2
10190 KM=4
10200 GOSUB15000
10210 IFERTHENRETURN
10220 T$=T0$
10230 GOSUB100
10240 TN$=T$
10250 TY$="CONW"
10260 TV=W
10270 GOSUB18000
10280 RETURN
10290 IFAD$=""THENER=19:GOSUB17000:AD$=CHR$(0)+CHR$(0)
10300 A$=":"
10310 GOSUB50
10320 IFA=0THEN10400
10330 TN$=LEFT$(T$,A-1)
10340 T$=MID$(T$,A+1)
10350 GOSUB150
10360 TY$="LABL"
10370 TV=AD
10380 GOSUB18000
10390 IFERTHENRETURN
10400 A$=" "
10410 AA$=T$
10420 GOSUB50
10430 IFA<>4ANDLEN(T$)>3THENER=1:GOSUB17000:RETURN
10440 IFT$=""THENRETURN
10450 A$=LEFT$(T$,3)
10460 GOSUB600
10470 IFK=0THENER=1:GOSUB17000:RETURN
10480 T$=MID$(T$,4)
10490 GOSUB14000
10500 IFERTHENRETURN
10510 IFMO>=0ANDKM=0THENER=7:GOSUB17000:RETURN
10520 IFMO=-1ANDKM>0THENER=3:GOSUB17000:RETURN
10530 ONKM+1GOTO11000,11100,11200,11300,11400,11500
10540 STOP
11000 M=0
11010 GOSUB12000
11020 RETURN
11100 M=2
11110 IFMO=1ORMO=4THENM=0
```

```
11120 IFM0=10THENM=1
11130 GOSUB12000
11140 HH=W
11150 GOSUB250
11160 C$=C$+RIGHT$(HH$,2)+LEFT$(HH$,2)
11170 RETURN
11200 M=M0
11210 GOSUB12000
11220 IFM0=9THENRETURN
11230 H1=(M0=0ORM0=1ORM0=2ORM0=3ORM0=7ORM0=8)
11240 IFH1THENH=W:GOSUB200:C$=C$+H$:RETURN
11250 HH=W
11260 GOSUB250
11270 C$=C$+RIGHT$(HH$,2)+LEFT$(HH$,2)
11280 RETURN
11300 IFW>255THENER=8:GOSUB17000:RETURN
11310 H=W
11320 GOSUB200
11330 C$=H$
11340 RETURN
11400 HH=W
11410 GOSUB250
11420 C$=RIGHT$(HH$,2)+LEFT$(HH$,2)
11430 RETURN
11500 M=1
11510 IFM0=1ORM0=4THENM=0
11520 GOSUB12000
11530 H=W-AD-2
11540 IFH>127ORH<-128THENER=17:GOSUB17000:RETURN
11550 IFH<0THENH=H+256
11560 GOSUB200
11570 C$=C$+H$
11580 RETURN
```

Mit diesem Unterprogramm wird eine Zeile des Source-Codes in den entsprechenden Maschinencode umgewandelt.

Bemerkung eliminieren

Zunächst werden einige Variablen vorbesetzt (C$, U$, ER) und die führenden Leerzeichen abgeschnitten. In den Zeilen 10050 bis 10090 wird eine Bemerkung gesucht. Die Bemerkung muß immer hinten stehen und wird anschließend abgeschnitten, da sie für die Assemblierung nicht gebraucht wird.

Direktive feststellen

Liegt eine Direktive vor, so wird das Unterprogramm zum Behandeln der Direktiven aufgerufen und das Unterprogramm zum Assemblieren einer Zeile sofort verlassen.

Konstanten-Definition

In den Zeilen 10110 bis 10280 wird die Definition von Konstanten behandelt. Dazu wird zunächst ein Gleichheitszeichen in der Source-Zeile gesucht und wenn kein Gleichheitszeichen vorliegt, wird zur Behandlung von Labels übergegangen (ab Zeile 10290). Zunächst werden die beiden Teilausdrücke vor und hinter dem Gleichheitszeichen in den Variablen T0$ und T$ zwischengespeichert und eventuell auftauchende Leerzeichen eliminiert. Dann werden die Variablen für die Größe des Symbols und den Modus des Befehls gesetzt, die bei einer Konstanten-Definition immer SI=2 und KM=5 sind. Da die Konstantenvereinbarung aus einem Doppelterm (zwei Terme verknüpft mit einem einzigen Operator) bestehen kann, wird das Unterprogramm zur Auswertung dieses Ausdrucks aufgerufen. Anschließend erfolgt eine Abprüfung, ob ein Fehler vorlag, wenn ja, wird das Unterprogramm abgebrochen. In den Zeilen 10220 bis 10270 wird das Symbol in die Symboltabelle eingefügt. Der im Unterprogramm ab Zeile 15000 berechnete Wert wird dem Symbol zugeordnet, der Typ des Symbols ist immer eine Konstante Word (CONW), und der Name wird in TN$ abgespeichert.

Labels / Marken

Vor der Behandlung der Marken wird zunächst abgeprüft, ob eine Startadresse gegeben war oder nicht. Wenn nicht, wird eine Fehlermeldung ausgegeben und die Startadresse auf 0 gesetzt.

Zeilen, die Labels / Marken beinhalten, enthalten auch immer einen Doppelpunkt. Zunächst wird abgeprüft ob die Zeile einen Doppelpunkt enthält, wenn nicht wird die Behandlung der Labels übersprungen.

Anschließend wird der Variablen TN$ der Name des Labels zugeordnet, der Variablen C$ eine eventuell folgende Befehlszeile. Anschließend wird - ähnlich den Konstanten - das Label in einer Tabelle eingeordnet.

Typ und Modus einer Befehlszeile feststellen

Wenn weder eine Direktive noch eine Konstante noch ein Label vorliegt, so muß zwangsläufig eine Befehlszeile vorliegen. Das Programmstück von Zeile 10400 bis 10540 stellt den Modus und den Typ einer Befehlszeile fest und verzweigt in Zeile 10530 aufgrund des festgestellten Typs in die entsprechenden Unterprogrammstücke.

Zunächst wird die gefundene mnemotechnische Bezeichnung in der Tabelle gesucht und aus den zugehörigen Werten Typ und Modus festgelegt.

Befehle ohne Operand

Befehle ohne Operanten sind am einfachsten zu handhaben. Für sie braucht lediglich der Code des eigentlichen Befehls übergeben zu werden, was durch ein Unterprogrammaufruf von Zeile 12000 erfolgt.

Absolute und indirekte Sprünge

Entsprechend der Sprungart wird die Variable M auf 0 (absolute Sprünge) oder auf 1 (indirekte Sprünge) gesetzt. Zeile 11130 setzt den entsprechenden Code in die Variable C$, und der Rest des Programmstückes bis hin zur Zeile 11160 wandelt den Operanden in eine hexadezimale Zahl um und ergänzt die Befehlszeile C$.

Befehle (keine Sprünge) mit Operanden

Analog den eben aufgezeigten Programmstücken wird auch hier wieder entsprechend dem Modus der hexadezimale Wert des Befehls in die Variable C$ geschrieben und anschliessend die in hexadezimale Zeichen umgerechneten Operanden.

Byte und Word

Die beiden Befehle BYT und WOR sind Konstanten-Definitionen für jeweils ein Byte bzw. zwei Byte. Diesen Konstanten wird zunächst keine Bezeichnung zugeordnet, sondern nur ein Wert. Z.B. werden auf diese Art und Weise Variablen definiert, mit z.B. 'VAR1: BYT 00'.

Relative Sprünge

In dem letzten Stück des Unterprogramms ab Zeile 11500 werden die relativen Sprünge behandelt, in dem auch hier wieder die Befehlszeile im Hexcode (C$) mit dem entsprechenden hexadezimalen Wert des Befehls und - als Operand - mit der Zahl der Bytes (in Hex-Code) besetzt wird, um die gesprungen werden soll. Einerseits dienen relative Sprünge der Rechengeschwindigkeit, andererseits wird die Symboltabelle nicht so groß, da nicht soviele absolute Adressen verwendet werden müssen. Ein weiterer Grund ist die Spei-

cherplatzersparnis, da relative Sprünge nur um maximal 127 Bytes nach vorne bzw. 128 Byte zurückspringen können, und damit der Operand in ein Byte paßt. Wurde für die relativen Sprünge eine Sprungweite größer als der eben angegebene Bereich errechnet, so wird eine Fehlermeldung ausgegeben.

ZEILE ASSEMBLIEREN **10000 - 11580**

Variablen:

Name	Typ	Bereich	Bedeutung
A	R	0...255	Position v. A$ in AA$
A$	P	1 Zeichen	zu suchendes Zeichen
AA$	P	Zeichenreihe	durchzusuchende Zeichenreihe
AD	G	0...65535	aktuelle Adresse
AD$	G	'' oder 2 Byte	Startadresse (L/H)
C$	A	bis 5 Zeichen	ass. Maschinencode
ER	A/P/R	0...E9	Fehlernummer
H1	H	0 oder -1	logische Hilfsvar.
HH	P	0...65535	Wert für Hexzahl
K	R/P	0...K9	Nummer des Befehls
KM	P	0...5	Typ des Befehls
M	P	0...10	Adressierungsmodus
MO	R	-1...10	Modus
SI	P	1 oder 2	Anzahl Byte des Oper.
T$	E/P	Zeichenreihe	zu assemblier. Zeile
TO$	H	Zeichenreihe	Zeichenreihe bis '='
TN$	P	Zeichenreihe	Symbolname
TV	P/R	0...65535	Wert des Symbols
TY$	P/R	'conw','labl',...	Typ des Symbols
U$	A	' ' oder 'R'	'R' wenn Operand noch nicht definiert
W	R	0...15535	Wert des Doppelterms

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
keine				

Unterprogrammaufrufe :

in	nach	Zweck
10040	150	Führende Blanks von T$ eliminieren
10070	50	Position v. A$ in AA$
10090	100	Nachf. Blanks von T$ eliminieren
10100	13000	Directiven auswerten
10130	50	Position v. A$ in AA$
10170	150	Führende Blanks von T$ eliminieren
10200	15000	Doppelterm auswerten
10230	100	Nachf. Blanks von T$ eliminieren
10270	18000	Symbol in Tabelle einfügen
10290	17000	Fehler registieren
10310	50	Position v. A$ in AA$
10350	150	Führende Blanks von T$ eliminieren
10380	18000	Symbol in Tabelle einfügen
10420	50	Position v. A$ in AA$
10430	17000	Fehler registieren
10460	600	Check auf Mnemonic Keyword
10470	17000	Fehler registieren
10490	14000	Mode feststellen
10510	17000	Fehler registieren
10520	17000	Fehler registieren
11010	12000	C$ mit Code besetzen
11130	12000	C$ mit Code besetzen
11150	250	4-stell. Hex-Zahl aus HH bilden
11210	12000	C$ mit Code besetzen
11240	200	2-stell. Hex-Zahl aus H bilden
11260	250	4-stell. Hex-Zahl aus HH bilden
11300	17000	Fehler registieren
11320	200	2-stell. Hex-Zahl aus H bilden
11410	250	4-stell. Hex-Zahl aus HH bilden
11520	12000	C$ mit Code besetzen
11540	17000	Fehler registieren
11560	200	2-stell. Hex-Zahl aus H bilden

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
10100	RETURN	LEFT$(T$,1)="%"	Direktive
10210	RETURN	ER NE 0	Fehler beim Doppel-
			term auswerten
10280	RETURN	'=' war in T$	Symbol eingefügt

```
!=========================================================!
! 10390 ! RETURN! ER NE 0             ! Fehler beim Ein-   !
!       !       !                     ! fügen eines Labels !
! 10430 ! RETURN! A NE 4  AND         ! Kein Blank nach    !
!       !       ! LEN(T$) GT 3        ! Befehl             !
! 10440 ! RETURN! T$=''               ! Zeile abgearbeitet !
! 10470 ! RETURN! K=0                 ! Ungültiger Befehl  !
! 10500 ! RETURN! ER NE 0             ! Fehler beim Fest-  !
!       !       !                     ! stellen des Modus  !
! 10510 ! RETURN! MO GE 0  AND        ! Wenn KM=0,dann kein!
!       !       ! KM = 0              ! Operand erlaubt    !
! 10520 ! RETURN! MO = -1  AND        ! Operand fehlt,     !
!       !       ! KM GT 0             ! obwohl nötig       !
! 10020 ! RETURN! KM = 0              ! kein Operand       !
! 10540 ! STOP  ! KM GT 5             ! Wert von KM unmögl.!
! 11170 ! RETURN! KM = 1              ! JMP oder JSR-Befehl!
! 11220 ! RETURN! KM = 2 AND MO = 9   ! Operand = Akku     !
! 11240 ! RETURN! H1                  ! Ein-Byte-Operand   !
! 11280 ! RETURN! KM = 2 AND NOT H1   ! Zwei-Byte-Operand  !
! 11300 ! RETURN! KM = 3   AND        ! Byte größer als 255!
!       !       ! W GT 255            !                    !
! 11340 ! RETURN! KM = 3              ! Byte reserviert    !
! 11430 ! RETURN! KM = 4              ! Word reserviert    !
! 11540 ! RETURN! H GT 127  OR        ! Relativer Sprung   !
!       !       ! H LT -128           ! zu weit            !
! 11580 ! RETURN! KM = 5              ! Relativer Sprung   !
!=========================================================!
```

5.3.2 Variable mit hexadezimalem Code besetzen

```
12010 C$=KC$(K,M)
12020 IFC$=""THENER=6:GOSUB17000
12030 RETURN
```

In diesem kurzen Unterprogramm wird in der Variablen C$ der hexadezimale Befehlscode eingetragen. Dazu wird aus den in Kapitel 5.1 beschriebenen DATA-Statements lediglich der entsprechende Code aus der Variabeln KC$ (K,M) übernommen. Wurde kein Code gefunden, so wird eine Fehlermeldung ausgedruckt.

5.3.3 Direktiven auswerten

```
13010 AA$=MID$(T$,2)
13020 GOSUB24000
```

```
13030 ONOGOTO13100,13200,13300
13040 ER=2
13050 GOSUB17000
13060 RETURN
13100 REM *** %ORG DIRECTIVE ***
13105 IFAD$>""THENER=20:GOSUB17000:RETURN
13110 T$=MID$(T$,5)
13115 KM=4
13120 GOSUB100
13125 GOSUB150
13130 GOSUB16000
13135 IFERTHENRETURN
13140 AD=W
13145 HH=INT(AD/256)
13150 H=AD-256*HH
13155 AD$=CHR$(H)+CHR$(HH)
13160 PRINT#3,AD$;
13165 SA=AD
13170 RETURN
13200 REM *** %END DIRECTIVE ***
13205 IS=320
13210 RETURN
13300 REM *** %INO DIECTIVE ***
13305 IFAD$>""THENER=20:GOSUB17000:RETURN
13310 KM=4
13315 INPUT"■STARTADRESSE    ■■■";T$
13320 IFT$=" "THEN13315
13325 GOSUB16000
13330 IFERTHENRETURN
13335 AD=W
13340 HH=INT(AD/256)
13345 H=AD-256*HH
13350 AD$=CHR$(H)+CHR$(HH)
13355 PRINT#3,AD$;
13360 SA=AD
13365 RETURN
```

Das Unterprogramm ist wie folgt unterteilt:

```
13000 Verteiler
13100 %ORG - absoluter Start des Programms
13200 %END - Ende des Sources-Listings
13300 %INO - Input der absoluten Startadresse vom
             Bildschirm
```

Zunächst wird in der Variablen AA$ der Name der Direktive (der auch mehr als drei Zeichen enthalten kann) abgelegt und mit dem Unterprogramm ab Zeile 24000 die Nummer der Direktive bestimmt. Aufgrund dieser Nummer wird anschliessend verzweigt. Wenn nicht verzweigt wird, so wird eine Fehlermeldung ausgegeben.

%ORG

Die ersten zwei Byte des Object-Files werden in der Variablen AD$ abgespeichert. Wenn schon eine Startadresse definiert wurde, so wird ein Fehler ausgegeben. Ansonsten wird in den Zeilen 13110 bis 13140 der Operand ausgewertet und in der Variablen AD abgelegt. In dem restlichen Programmstück wird die Dezimalzahl noch in zwei Characters zerlegt, die jeweils das Lower und Higher Byte angeben. An dieser Stelle sei noch angemerkt, daß das Abspeichern in der Object-Datei (auf Floppy) jeweils in zwei zusammengefaßten Hexadezimalziffern zu einem Zeichen gespeichert wird. Z.B. wird hexadezimal 'FF' als CHR$ (255) gespeichert.

Als Abschluß wird noch in der globalen Variablen SA der Wert der Startadresse festgehalten, um ihn im weiteren Programm verwenden zu können.

%END

In dieser Zeile wird nur der INPUT-Status als Merker für das Ende des Sources-Listings gesetzt.

%INO

Dieses Programmstück funktioniert analog der in den Zeilen 13100 bis 13170 beschriebenen % ORG - Direktive, außer daß die Startadresse nicht als Operand im Source-Listing steht, sondern vom Bildschirm eingelesen wird.

DIREKTIVEN AUSWERTEN 13000 - 13365

Variablen:

Name	Typ	Bereich	Bedeutung
AA$	P	Zeichenreihe	in der gesucht wird
AD	G	0...65535	aktuelle Adresse
AD$	G	2 Zeichen (Byte)	Startadresse mit CHR$
ER	P	0..E9	Fehlernummer

H	H	0...255	Lower-Byte Startadr.
HH	H	0...255	Higher-Byte Startadr.
IS	A	0...320	Input-Status
KM	P	0...5	Typ des Befehls
SA	G	0...65535	Startadresse
T$	P	Zeichenreihe	Vom Bildschirm eingelesene Startadresse

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
keine				

Dateien :

#	Name	T	Bemerkung
3	F$+"...."	p	Vorläufige Objektdatei

Unterprogrammaufrufe :

in	nach	Zweck
13020	24000	Nummer der Direktive feststellen
13050	17000	Fehler registrieren
13105	17000	Fehler registrieren
13120	100	Nachfolgende Blanks eliminieren
13125	150	Führende Blanks eliminieren
13130	16000	Einzelausdruck auswerten
13305	17000	Fehler registrieren
13325	16000	Einzelausdruck auswerten

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
13060	RETURN	D=0	keine Direktive
13105	RETURN	AD$ GT ""	Startadresse bereits definiert
13135	RETURN	ER NE 0	Fehler im Einzel-

```
!=========================================================!
!       !       !                       ! term            !
! 13170 ! RETURN! D=1                   ! Ende von %org   !
! 13210 ! RETURN! D=2                   ! Ende von %end   !
! 13305 ! RETURN! AD$ GT ""             ! Startadresse    !
!       !       !                       ! bereits definiert !
! 13330 ! RETURN! ER NE 0               ! Fehler im Einzel- !
!       !       !                       ! term            !
! 13365 ! RETURN! D=3                   ! Ende von %ino   !
!=========================================================!
```

5.3.4 Mode feststellen

```
14030 REM MODE MO : -1=NO OPER.;  0=# USW. S.ZEILE 54030
14040 REM MODE MO = 10 : INDIRECT
14050 REM SIZE:  SI=1 -> BYTE     SI=2 -> WORD
14060 GOSUB150
14070 IFT$=""THENMO=-1:RETURN
14080 IFT$="A"THENMO=9:RETURN
14090 AA$=T$
14100 FORSZ=1TO7
14110 A$=MID$(SZ$,SZ,1)
14120 GOSUB50
14130 A(SZ)=A
14140 NEXTSZ
14150 IFA(2)<>-(A(3)>0)THENER=4:GOTO17000
14160 L=LEN(T$)
14170 XY=(RIGHT$(T$,1)<>"X"ANDRIGHT$(T$,1)<>"Y")
14180 IFA(4)THENIFA(5)>0ORA(4)<>L-1ORXYTHEN14500
14190 IFA(5)THENIFA(4)>0ORA(5)<>L-1ORXYTHEN14500
14200 IFA(1)>1ORA(6)>1ORA(7)>1THENER=5:GOTO17000
14210 IFA(1)THENIFA(2)+A(3)+A(4)+A(5)+A(6)+A(7)THEN14500
14220 IFA(6)THENIFA(7)THEN14500
14230 IFA(7)THENIFA(6)THEN14500
14240 IFA(6)+A(7)THENIFA(1)+A(2)+A(3)THEN14500
14250 IFA(1)THENMO=0:T$=MID$(T$,2):GOTO14610
14260 IFRIGHT$(T$,3)=",X)"THENMO=7:GOTO14600
14270 IFRIGHT$(T$,3)=".X)"THENMO=7:GOTO14600
14280 IFRIGHT$(T$,3)="),Y"THENMO=8:GOTO14600
14290 IFRIGHT$(T$,3)=").Y"THENMO=8:GOTO14600
14300 IFRIGHT$(T$,1)=")"THENMO=10:T$=MID$(T$,2,L-2):
                                    SI=2:GOSUB15000:RETURN
14310 ZP=0
14320 SI=2
14330 IFA(6)THENZP=1:T$=MID$(T$,2):L=L-1:SI=1
14340 IFA(7)THENZP=2:T$=MID$(T$,2):L=L-1
14350 IFRIGHT$(T$,2)=",X"THENMO=5:GOSUB14440:GOTO14410
14360 IFRIGHT$(T$,2)=".X"THENMO=5:GOSUB14440:GOTO14410
14370 IFRIGHT$(T$,2)=",Y"THENMO=6:GOSUB14440:GOTO14410
14380 IFRIGHT$(T$,2)=".Y"THENMO=6:GOSUB14440:GOTO14410
14390 MO=4
14400 GOSUB14450
```

```
14410 IFERTHENRETURN
14420 IFW<256ANDZP<2THENMO=MO-3
14430 RETURN
14440 T$=LEFT$(T$,L-2)
14450 IFZP=1THENMO=MO-3
14460 GOSUB15000
14470 RETURN
14500 ER=10
14510 GOSUB17000
14520 RETURN
14600 T$=MID$(T$,2,L-4)
14610 SI=1
14620 GOSUB15000
14630 RETURN
```

Dieses Unterprogramm dient letztendlich zum Besetzen der Variablen MO, in der der Adressmodus des jeweiligen Befehls festgelegt wird. Nicht festgestellt wird in diesem Unterprogramm, ob der jeweilige Befehl auch diesen Modus kennt. Aufgrund der Sonderzeichen und der Größe der Operanden wird dieser Modus festgestellt, z.B. wenn der Befehl ein ',Y' enthält, kann nur noch der Mode 3, 6 oder 8 vorliegen, und liegen z.B. auch noch Klammern vor, so steht fest, daß der Modus 8 für einen indirekt indizierter adressierten Befehl vorliegt.

Ist der Operand eine leere Zeichenreihe, so wird MO mit '1' besetzt , ist der Operand ein 'A' wird MO mit '9' besetzt. In beiden Fällen wird das Unterprogramm sofort verlassen.

Das Feststellen der Sonderzeichen und Besetzen des Arrays A() geschieht in den Zeilen 14060 bis 14140. Die Zeilen 14150 bis 14240 enthalten diverse Prüfungen auf die Richtigkeit der Syntax (sind Klammern richtig gesetzt,...).

In den Zeilen 14250 bis 14300 werden Modes behandelt, die aufgrund der vorhergehenden Programmzeilen schon definiert werden können. In den restlichen Zeilen wird jeweils für den Modus noch entschieden, ob die Adressierungsart in die Zeropage hineinzeigt, oder ob der Operand eine absolute Adresse angibt.

Mode feststellen 14000 - 14630

Variablen:

Name	Typ	Bereich	Bedeutung
A	R	0...255	Position v. A$ in AA$
A$	P	1 Zeichen	zu suchendes Zeichen
AA$	P	Zeichenreihe	durchzusuchende Zei.r.
ER	P/R	0...£9	Fehlernummer
L	H	0...79	Länge von T$
MO	A	-1...10	Mode
			-1 = kein Operand
			0 = immediate
			1 = zeropage
			2 = zeropage,x
			3 = zeropage,y
			4 = absolute
			5 = absolute,x
			6 = absolute,y
			7 = indexed indirect
			8 = indirect indexed
			9 = accu
			10 = indirect
SI	A	1 oder 2	Anzahl Bytes des Operanden
SZ	H	1...7	Nummer des zu suchenden Sonderzeichens
SZ$	G	"#(),.§!+-*/$'"	alle Sonderzeichen
T$	E/P/A	Zeichenreihe	Operand
W	R	0...65535	Wert des Doppelterms
XY	H	0 oder -1	XY=0,wenn letztes Zei. d. Oper. 'x' od. 'y'
ZP	H	0...2	Zeropage-Option
			0=keine Vorgabe
			1=Vorgabe : Zeropage
			2=Vorgabe : Absolute

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
A	LEN(SZ$)	H	0...255	Position jedes Sonderz. in AA$

Unterprogrammaufrufe :

in	nach	Zweck
14060	150	Führende Blanks von T$ eliminieren
14120	50	Position v. A$ in AA$ bestimmen
14300	15000	Doppelausdruck auswerten
14460	15000	Doppelausdruck auswerten
14510	17000	Fehler registrieren
14620	15000	Doppelausdruck auswerten

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
14070	RETURN	T$=""	Kein Operand
14080	RETURN	T$="a"	Operand = Akku
14150	17000	a(2) NE -(a(3) GT 0)	Klammern nicht paarweise
14200	17000	a(1) GT 1 OR a(6) GT 1 OR a(7) GT 1	Immediate- bzw. Zeropagezeichen nicht an 1.Stelle
14300	RETURN	RIGHT$(T$,1)=")"	indirect adressing
14410	RETURN	ER NE 0	Fehler
14430	RETURN		Ende
14520	RETURN	div. Möglichkeiten	Fehlernummer 10
14630	RETURN		Ende

5.3.5 Doppelterm auswerten

```
15010 AA$=T$
15020 FORSZ=1TO11
15030 A$=MID$(SZ$,SZ,1)
15040 GOSUB50
```

```
15050 A(SZ)=A
15060 NEXT
15070 IFA(1)+A(2)+A(3)+A(4)+A(5)+A(6)+A(7)THEN15500
15080 A=-(A(8)>0)-(A(9)>0)-(A(10)>0)-(A(11)>0)
15090 IFA=0THENGOSUB16000:GOTO15250
15100 IFA>1THENER=12:GOTO17000
15110 A=A(8)+A(9)+A(10)+A(11)
15120 T1$=LEFT$(T$,A-1)
15130 T2$=MID$(T$,A+1)
15140 T$=T1$
15150 GOSUB16000
15160 W1=W
15170 T$=T2$
15180 GOSUB16000
15190 W2=W
15200 IFERTHENRETURN
15210 IFA(8)THENW=W1+W2
15220 IFA(9)THENW=W1-W2
15230 IFA(10)THENW=W1*W2
15240 IFA(11)THENW=INT(W1/W2)
15250 IFW>=256↑SITHENER=13:GOTO17000
15260 RETURN
15500 ER=10
15510 GOSUB17000
15520 RETURN
```

Unter Doppelterm verstehen wir bei dem Assembler Terme, die aus zwei Operanden und einem Verknüpfungs-Operator wie z.B. '+' bestehen. In diesem Unterprogramm werden nur Doppelterme ausgewertet. Jedoch werden auch die Einzelauswertungen hier behandelt, da dann das Unterprogramm ab Zeile 16000 in 15150 aufgerufen wird.

Die Zeilen 15010 bis 15060 beinhalten wieder eine Prüfung auf Sonderzeichen. Wenn irgendeines der ersten sieben Sonderzeichen wie sie in dem Kasten bei Mode feststellen unter der Variablen SZ$ festgehalten sind, hier auftreten, wird zur Zeile 15500 übergegangen, wo eine Fehlermeldung ausgedruckt wird.

In Zeile 15080 wird in der Variablen A aufsummiert, wieviele der Sonderzeichen '+','-','*' und '/', in der Befehlszeile erscheinen. Wenn kein Sonderzeichen erscheint, liegt ein Einzelausdruck vor, und das entsprechende Unterprogramm wird aufgerufen und zum Ende des Unterprogramms verzweigt.

Wenn mehr als ein Sonderzeichen vorliegt, wird eine Fehlermeldung ausgegeben. In Zeile 15110 wird der Variablen A die Position des Sonderzeichens zugewiesen. Die Variablen

T1$ (linker Teil) und T2$ (rechter Teil) beinhalten anschließend die durch das Sonderzeichen getrennten Textteile. Beide Ausdrücke werden durch Aufrufen des Unterprogramms für Einzelausdruck auswerten getrennt behandelt.

In den Zeilen 15210 bis 15240 werden die eigentlichen Verknüpfungen durchgeführt, wobei die in den Variablen W1 und W2 festgehaltenen Dezimalwerte der Operanden verwendet werden.

Doppelausdruck auswerten 15000 - 15520

Variablen:

Name	Typ	Bereich	Bedeutung
A	R/H	0...255	Position v. A$ in AA$
A$	P	1 Zeichen	zu suchendes Zeichen
AA$	P	Zeichenreihe	durchzusuchende Zei.r.
ER	P/R	0...E9	Fehlernummer
SI	E	1 oder 2	Anzahl Bytes des Operanden
SZ	H	1...7	Nummer des zu suchenden Sonderzeichens
SZ$	G	"#(),.§!+-*/$'"	alle Sonderzeichen
T$	E/P/A	Zeichenreihe	Operand
T1$	H	Zeichenreihe	Erster Term
T2$	H	Zeichenreihe	Zweiter Term
W	R/A	0...65535	Wert des Einzelterms / Wert des Doppelterms
W1	H	0...65535	Wert des 1.Terms
W2	H	0...65535	Wert des 2.Terms

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
A	LEN(SZ$)	H	0...255	Position jedes Sonderz. in AA$

Unterprogrammaufrufe :		
in	nach	Zweck
15040	50	Position von A$ in AA$ bestimmen
15090	16000	Einzelausdruck auswerten
15150	16000	Einzelausdruck auswerten
15180	16000	Einzelausdruck auswerten
15510	17000	Fehler registrieren

Verzweigungen nach außen :			
in Ze	nach	Bedingung	Bemerkung
15100	17000	A GT 1	Mehr als 1 Operator
15200	RETURN	ER NE 0	Fehler
15260	RETURN		Ende
15500	RETURN	A(1...7) NE 0	Falsches Sonderz. im Doppelterm

5.3.6 Einzelausdruck auswerten

Zunächst werden in dem Unterprogramm zum Auswerten eines Einzelausdrucks auch wieder die führenden und folgenden Leerzeichen abgeschnitten. Wie bereits im letzten Kapitel erwähnt, wird dieses Unterprogramm bei Doppeltermen auch ausgewertet, dann jedoch für jeden Teil des Terms getrennt.

Wenn der Einzelterm nur aus einem '$' besteht, so wird die Adresse der eigenen Assemblerzeile als Wert herangezogen. Liegen hinter dem '$' noch mehr Zeichen vor, so steht dort entweder ein hexadezimaler oder ein binärer Ausdruck. Ob hexadezimal oder binär, ergibt sich aus der Länge des dem $-Zeichen folgenden Ausdruckes: acht weitere Zeichen bedeuten immer binär; zwei oder vier Zeichen immer hexadezimal.

Wenn der Operand eine Ziffer ist, so wird dessen Wert einfach über die Funktion VAL bestimmt. In allen anderen Fällen, liegt mit Sicherheit ein Symbol vor, das dann im weiteren in der Symboltabelle herausgesucht wird. Wenn das Symbol in der Tabelle gefunden wurde, so wird der dort ge-

```
16010 GOSUB100
16020 GOSUB150
16030 IFT$="$"THENW=AD:RETURN
16040 IFLEFT$(T$,1)<>"$"THEN16120
16050 T$=MID$(T$,2)
16060 IFLEN(T$)=2THENH$=T$:GOSUB350:W=H:RETURN
16070 IFLEN(T$)=4THENHH$=T$:GOSUB400:W=HH:RETURN
16080 IFLEN(T$)=8THENBB$=T$:GOSUB23000:W=BB:RETURN
16090 ER=14
16100 GOSUB17000
16110 RETURN
16120 IFASC(T$)>=48ANDASC(T$)<=57THENW=VAL(T$):RETURN
16130 TN$=T$
16140 GOSUB500
16150 IFTFTHENW=TV:RETURN
16160 UA=AD+1
16170 UN$=LEFT$(T$,8)
16180 ONKMGOTO16230,16290,16370,16400,16430
16190 ER=16
16200 GOSUB17000
16210 UT=0
16220 GOTO16450
16230 IFMO=10RMO=4THENUT=7:GOTO16450
16240 IFMO=10THENUT=6:GOTO16450
16250 ER=6
16260 GOSUB17000
16270 UT=0
16280 GOTO16450
16290 IFMO=0THENUT=1:GOTO16450
16300 IFMO<4THENUT=3:GOTO16450
16310 IFMO<=6THENUT=5:GOTO16450
16320 IFMO<=8THENUT=4:GOTO16450
16330 ER=6
16340 GOSUB17000
16350 UT=0
16360 GOTO16450
16370 UT=1
16380 UA=AD
16390 GOTO16450
16400 UT=2
16410 UA=AD
16420 GOTO16450
16430 UT=3
16440 GOTO16450
16450 GOSUB19000
16460 U$="R"
16470 W=32768+128:REM DEFAULT WERT FUER ADRESSEN &8080
16480 IFUT=1ORUT=3ORUT=4THENW=128:REM DEFWERT FUER BYTEW.
16490 IFUT=8THENW=AD+2:REM DEFWERT FUER BRANCHES
16500 RETURN
```

speicherte Wert einfach der Variablen W zugeordnet, und das Unterprogramm verlassen.

In den Zeilen 16160 bis 16500 erfolgt die Behandlung für undefinierte Symbole (im allgemeinen Vorwärtssprünge), die in eine Tabelle für undefinierte Symbole eingetragen werden. In der Variablen UA wird die Adresse des undefinierten Symbols festgehalten und in der Variablen UN$ die ersten acht Zeichen des Symbols. Im folgenden muß nur noch der Typ des undefinierten Symbols bestimmt werden. Abhängig vom Typ der vorangegangenen Anweisung wird in Zeile 16180 nach verschiedenen Programmstücken verzweigt.

Diese Zeilen befinden sich insgesamt im Bereich 16230 bis 16440. Hier wird abhängig vom Typ der Anweisung und vom Mode der Anweisung die Variable UT besetzt. In den Zeilen 16450 bis 16500 wird schließlich das undefinierte Symbol in die Tabelle der undefinierten Symbole eingetragen, die Variable U$ auf 'R' als Merker gesetzt, und die Variable U mit einem Default-Wert, abhängig vom Typ des undefinierten Symbols, besetzt.

Einzelausdruck auswerten 16000 - 16500

Variablen:

Name	Typ	Bereich	Bedeutung
AD	G	0...65535	aktuelle Adresse
BB	R	0...255	Wert von Binär-Zahl
BB$	P	Zeichenreihe	Binär-Zahl
ER	P/R	0...E9	Fehlernummer
H	R	0...255	Wert v. 2-stell. Hex-Z
H$	P	2 Zeichen	2-stellige Hex-Zahl
HH	R	0...65535	Wert v. 4-stell. Hex-Z
HH$	P	4 Zeichen	4-stellige Hex-Zahl
KM	E	0...5	Typ des Befehls
MO	E	-1...10	Adressierungsmodus
T$	E	Zeichenreihe	Einzelterm
TF	R	0...300	Nummer des Symbols
TN$	P	Zeichenreihe	Gesuchtes Symbol
TV	R	0...65535	Wert des Symbols
U$	A	' ' oder 'R'	'R' bei undef. Symbol
UA	P	0...65535	Adresse, bei der un-

			def. Symbol auftrat
UN$	P	Zeichenreihe	Name des und. Symbols
UT	P	0...8	Typ des undef. Symbols
W	A	0...65535	Wert des Einzelterms

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung

Unterprogrammaufrufe :

in	nach	Zweck
16010	100	nachfolgende Blanks eliminieren
16020	150	führende Blanks eliminieren
16060	350	Wert von 2-stell. Hex-Zahl bestimmen
16070	400	Wert von 4-stell. Hex-Zahl bestimmen
16080	23000	Wert von 8-stell. Binärzahl bestimmen
16100	17000	Fehler registrieren
16140	500	Symbol in Tabelle suchen
16200	17000	Fehler registrieren
16260	17000	Fehler registrieren
16340	17000	Fehler registrieren
16450	19000	Undefiniertes Symbol in Tabelle eintr.

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
16030	RETURN	T$ = '$'	Term = '$'
16060	RETURN	LEN(T$)=2	Term=2-stell. Hex-Z
16070	RETURN	LEN(T$)=4	Term=4-stell. Hex-Z
16080	RETURN	LEN(T$)=8	Term=8-stell. Bin-Z
16110	RETURN	...	Falsche Länge $Term
16120	RETURN	T$ beginnt m. Ziff	Term aus Ziffern
16500	RETURN		normales Ende

5.3.7 Weitere Unterprogramme

Fehler registrieren

```
17010 EN=EN+1
17020 EB=EB+1
17030 IFEB>10THENEB=10:ER=18
17040 EB(EB)=ER
17050 RETURN
```

In diesem kurzen Unterprogramm wird ein Fehler im Quelltext, der an verschiedenen Stellen des Assembler-Programms entdeckt werden kann, registriert. Die Fehler werden später gesammelt unter **die Zeile** des Quelltextes geschrieben, in der die Fehler aufgetaucht sind.

Zunächst werden die beiden Variablen EN (für die Anzahl der Fehler im gesamten Programm) und EB (für die Anzahl der Fehler in dieser Quellzeile) jeweils um eins erhöht. Wenn die Anzahl der Fehler in einer Zeile größer als zehn ist, wird anstatt des eigentlichen Fehlers die Meldung 'zu viele Fehler' gespeichert. Die Speicherung erfolgt in dem Feld EB().

Neues Symbol in Tabelle eintragen

```
18010 GOSUB500
18020 IFTFTHENER=11:GOSUB17000:RETURN
18030 TA=TA+1
18040 TY$(TA)=TY$
18050 TV(TA)=TV
18060 TN$(TA)=TN$
18070 RETURN
```

Dieses Unterprogramm dient zum Eintragen neu auftauchender Symbole oder Labels (Marken). Ist das Symbol bereits in der Tabelle vorhanden, was durch den Aufruf des Unterprogrammes ab Zeile 500 festgestellt wird, so wird eine Fehlermeldung gespeichert und das Unterprogramm verlassen. Ansonsten wird die Variable TA um eins erhöht, die angibt wieviele Symbole in der Tabelle gespeichert sind, und in den Feldern TY$(), TV(), und TN$() werden der Typ, der Wert und der Name des Symbols gespeichert.

Wert in Tabelle der undefinierten Symbole eintragen

```
19010 U=U+1
19020 UT%(U)=UT
19030 UA(U)=UA
19040 UN$(U)=UN$
19050 RETURN
```

Wird in einem Assembler-Programm ein Symbol angesprochen, das zu diesem Zeitpunkt noch nicht definiert ist, so muß sich der Assembler die Adresse und den Typ des Symbols merken, damit er später, wenn das Symbol definiert ist, an der entsprechenden Stelle den Wert nachträglich eintragen kann. Dazu dient dieses kurze Unterprogramm.

Die Variable U, die die Anzahl der undefinierten Symbole angibt, wird dazu um eins erhöht. Das Feld UT%() beinhaltet den Typ der undefinierten Symbole, das Feld UA() die Adresse und das Feld UN$() den Namen der undefinierten Symbole.

Symbol-Tabelle speichern und drucken

```
20010 PRINT#4
20020 PRINT#4,"SYMBOLE:"
20030 PRINT#4,"NAME       TYP    WERT"
20035 PRINT#4,"-----------------------"
20040 OPEN5,8,5,"@:"+F$+".SYM,S,W"
20050 GOSUB25000
20060 IFDSTHENPRINTDS$:STOP
20070 IFTA=0THEN20150
20080 FORI=1TOTA
20090 PRINT#5,TY$(I);",";TN$(I);",";TV(I)
20100 PRINT#4,LEFT$(TN$(I)+SP$,10)LEFT$(TY$(I)+SP$,6);
20110 HH=TV(I)
20120 GOSUB250
20130 PRINT#4,HH$
20140 NEXTI
20150 CLOSE5
20160 PRINT#4
20170 RETURN
```

Wenn das Quellprogramm fertig assembliert ist, so ist auch die Symboltabelle komplett. Dieses kurze Unterprogramm gibt nun die gesamte Symboltabelle auf dem Drucker bzw.

auf dem Bildschirm aus und speichert sie zusätzlich in einer sequentiellen Datei auf Floppy.

In Zeile 20040 wird dazu die Datei #5 auf der Floppy mit dem Dateinamen F$+".SYM" zum Schreiben eröffnet. In einer Schleife (Zeilen 20080 bis 20140) werden alle Symbole nacheinander sowohl auf Floppy als auch auf Drucker ausgegeben. Auf Floppy werden die Symbolattribute Typ, Name und Wert durch Komma getrennt abgespeichert, auf den Drucker wird erst der Name, dann der Typ, und schließlich der Wert in hexadezimaler Form ausgegeben. Nach Beendigung der Schleife wird die Floppydatei geschlossen und das Unterprogramm verlassen.

Manuelle Eingabe von Symbolwerten

```
21010 PRINTTN$;
21020 INPUTT$
21030 ER=0
21040 GOSUB16000
21050 IFERTHEN21000
21060 IFUT=1THENTY$="CONB":SI=1
21070 IFUT=2THENTY$="CONW":SI=2
21080 IFUT=3ORUT=4THENTY$="ADRZ":SI=1
21090 IFUT=5ORUT=6THENTY$="ADRA":SI=2
21100 IFUT=7ORUT=8THENTY$="LABL":SI=2
21110 IFWD>=256↑SITHEN21000
21120 TV=W
21130 GOSUB13000
21140 RETURN
```

Wenn der Assembler beim nachträglichen Einsetzen der vorher undefinierten Symbole feststellt, daß ein bestimmtes Symbol immer noch nicht definiert ist, müßte eigentlich eine Fehlermeldung 'Symbol nicht definiert' ausgegeben werden. In dem vorliegenden Assembler wurde jedoch eine andere Möglichkeit gewählt. Ein Symbol das nicht definiert wurde, wird am Bildschirm erfragt. Dadurch erspart man sich den erneuten Lauf des Assemblers durch das gesamte Programm.

Zur Feststellung des Wertes wird vom Bildschirm die Zeichenreihe T$ eingelesen und sodann dem Unterprogramm 16000 übergeben, das den Wert eines Einzelausdruckes bestimmt. Dadurch kann die Eingabe auch hexadezimal oder binär erfolgen. War die Eingabe mit Fehlern behaftet, so wird

nochmal nach dem Symbol gefragt. Der Typ des neuen Symbols (TY$) richtet sich nach dem Typ des undefinierten Symbols (UT). In Zeile 21130 wird schließlich das neue Symbol in die Symboltabelle eingetragen und das Unterprogramm verlassen.

Eine Zeile einlesen

```
22010 IFTY=1THENINPUT#2,T$:IS=ST:ZN=ZN+1:GOTO22130
22020 GET#2,A1$,A2$:REM POINTER UEBERLESEN
22030 IFA1$=""ANDA2$=""THENIS=64:RETURN
22040 GET#2,A1$,A2$:REM ZEILENNUMMER LESEN
22050 ZN=0
22060 IFA1$<>""THENZN=ASC(A1$)
22070 IFA2$<>""THENZN=ZN+256*ASC(A2$)
22080 T$=""
22090 GET#2,A$
22100 IFA$=""THEN22130
22110 T$=T$+BK$(ASC(A$))
22120 GOTO22090
22130 ZN$=RIGHT$("      "+STR$(ZN),5)
22140 RETURN
```

Dieses Unterprogramm liest eine Zeile der Quelldatei in die Variable T$ ein.

Es wird unterschieden, ob die Eingabedatei eine sequentielle Datei ist, die eventuell mit einem Texteditor erstellt worden ist, oder eine Programmdatei, die mit dem normalen Editor erstellt worden ist. Diese Unterscheidung wird in der Variablen TY festgehalten. Bei TY=1 ist die Eingabedatei eine sequentielle Datei, und das Einlesen ist durch einen INPUT-Befehl zu realisieren. In der Variablen IS wird der Status der Eingabedatei festgehalten. IS=64 zeigt das Ende der Quelldatei an.

Im Fall der sequentiellen Eingabedatei sind ja keine Zeilennummern im Quelltext vorhanden, so daß die Zeilen fortlaufend durchnumeriert werden. Wenn die Eingabedatei eine Programmdatei ist, ist das Einlesen einer Zeile etwas komplizierter; zuerst müssen nämlich die ersten zwei Byte einer Zeile überlesen werden, die normalerweise den Zeiger auf die nächste Programmzeile darstellen. Sind diese beiden Zeiger jeweils '0', was sich dadurch bemerkbar macht, daß die beiden eingelesenen Zeichenreihen (A1$, A2$) 'leer' sind, so wird die Variable IS ebenfalls auf 64 gesetzt, um das Ende des Programms anzuzeigen und anschlies-

send das Unterprogramm verlassen.

Ist das Ende der Datei noch nicht erreicht, werden noch einmal zwei Byte von der Quelldatei eingelesen, die das niederwertige und höherwertige Byte der Zeilennummer darstellen. Diese Bytes werden in den Zeilen 22050 bis 22070 in die Variable ZN umgewandelt. Im Folgenden werden nun immer wieder einzelne Zeichen (A$) eingelesen, bis eine leere Zeichenreihe auftritt. Die Zeichen müssen aber noch dekodiert werden, da ein im Quelltext vorhandener Basic-Befehl als Sonderzeichen abgespeichert wird und deshalb hier durch den gesamten Text des Befehls ersetzt werden muß.

Nachdem in Zeile 20130 die Variable ZN$ mit einer fünfstelligen Zeichenreihe, gebildet aus der Zeilennummer, besetzt wurde, wird das Unterprogramm verlassen.

Bestimmung des Wertes einer Binärzahl

```
23010 BB=0
23020 FORI=1TOLEN(BB$)
23030 B=ASC(MID$(BB$,I))-48
23040 IFB<0ORB>1THENER=15:GOSUB17000:RETURN
23050 BB=2*BB+B
23060 NEXT
23070 RETURN
```

Dieses Unterprogramm legt den Wert einer Binärzahl, die als Zeichenreihe BB$ übergeben wird, in der Variablen BB ab.

Die Berechnung des Wertes geschieht in einer Schleife, indem der Hilfsvariablen B zunächst der Wert einer einzelnen Stelle der Binärzahl zugeordnet wird. Die Variable BB erhält man nun, indem man jeweils den alten Wert der Variablen BB verdoppelt und den Wert der einzelnen Stelle hinzuaddiert. Man kann sich überlegen, daß durch dieses Verfahren - das den Mathematikern als Horner-Schema bekannt ist - am Ende der Schleife der richtige Wert der Variablen BB steht.

Nummer der Direktive bestimmen

```
24010 FORD=1TOD9
24020 A$=D$(D)
24030 GOSUB50
24040 IFA=0THENNEXT
24050 IFD>D9THEND=0
24060 RETURN
```

Dieses Unteprogramm durchsucht die Variable AA$ nach einer vorhandenen Direktive.

Dazu wird jede mögliche Direktive mit Hilfe des Unterprogramms in Zeile 50 untersucht und anschließend die Nummer der eventuell gefundenen Direktive in der Variablen D zurückgegeben.

Disk-Status-Werte DS und DS$ bestimmen

```
25010 INPUT#15,DS,D1$,D2$,D3$
25020 DS$=STR$(DS)+","+D1$+","+D2$+","+D3$
25030 RETURN
```

Hier wird einfach der Status der Floppy aus deren Fehlerkanal in die Variablen DS und DS$ übertragen.

Die Bezeichnung der Ausgabevariablen wurde in Anlehnung an Basic 4.0 der Commodore 8000er Serie gewählt.

Weitere Unterprogramme	**17000 - 25030**
Fehler registrieren	17000 - 17050
Symbol in Tabelle eintragen	18000 - 18070
Undef. Symbol in Tab. eintr.	19000 - 19050
Symboltabelle ausgeben	20000 - 20170

Manuelle Eingabe von Symbolen	21000 - 21140
Eine Zeile einlesen	22000 - 22140
Wert einer Binärzahl bestimmen	23000 - 23070
Nummer der Direktive bestimmen	24000 - 24060
Disk-Status-Werte bestimmen	25000 - 25030

Variablen:

Name	Typ	Bereich	Bedeutung
EB	G	0...10	Anzahl Fehler in Zeile
EN	G	Ganzzahlig	Anzahl Fehler insges.
ER	E/A	0...E9	Fehlernummer
TA	G	0...300	Anzahl Symbole in Tab.
TF	R	0...300	Nummer des Symbols
TN$	E	Zeichenreihe	Gesuchtes Symbol
TV	E/A	0..65535	Wert des Symbols
TY$	E/A	Zeichenreihe	Typ des Symbols
U	G	0...300	Anz. undef. Symbole
UA	E	0...65535	Adresse, bei der un-def. Symbol auftrat
UN$	E	Zeichenreihe	Name des und. Symbols
UT	E	0...8	Typ des undef. Symbols
F$	G	Zeichenreihe	Dateiname
HH	P	0...65535	Wert v. 4-stell. Hex-Z
HH$	R	4 Zeichen	4-stellige Hex-Zahl
I	H	0...300	Laufvariable
SP$	G	ca. 50 Blanks	z. Auffüll. v. Strings
T$	P	Zeichenreihe	Einzelterm
SI	A	0...2	Anz. Byte im Operand
W	A	0...65535	Wert des Einzelterms
A$	H	1 Zeichen	Eingelesenes Zeichen
A1$	H	1 Zeichen	Eingelesenes Zeichen
A2$	H	1 Zeichen	Eingelesenes Zeichen
IS	A	0...320	Einlese-Status
TY	E	1 oder 2	Typ des Source-File
ZN	A	0...65535	Zeilennummer
ZN$	A	5 Zeichen	Zeilennr. als String
B	H	0...1	Eine Stelle d. Bin.Z.
BB	R	0...255	Wert von Binär-Zahl

BB$	P	Zeichenreihe	Binär-Zahl
A	R	0...255	Position v. A$ in AA$
D	A	0...D9	Nummer der Direktive
D1$	H	Zeichenreihe	Disk-Status (Klartext)
D2$	H	2 Zeichen	Disk-Status (Spurnr.)
D3$	H	2 Zeichen	Disk-Status (Sektor)
DS	A	0...99	Disk-Status (Nummer)
DS$	A	Zeichenreihe	DS,D1$,D2$,D3$ kombin.

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
BK$	0...203	G	Zeichenreihen	Basic-Befehle
D$	1...D9	E	Zeichenreihen	Direktiven
EB	1...10	A	0...E9	Fehlernummern
TN$	1...300	G	Zeichenreihen	Symbole
TV	1...300	G	0...65535	Werte d. Symbole
TY$	1...300	G	4 Zeichen	Typen d. Symbole
UA	1...300	G	0...65535	Adressen der undef. Symbole
UN$	1...300	G	Zeichenreihen	Namen der undef. Symbole
UT%	1...300	G	0...8	Typ d. und. Sym.

Dateien :

#	Name	T	Bemerkung
5	F$+".SYM"	S	Symboltabelle
15	Floppy		Fehlerkanal der Floppy

Unterprogrammaufrufe :

in	nach	Zweck
18010	500	Symbol in Tabelle suchen
20050	25000	Disk-Status bestimmen
20120	250	4-Stellige Hex-Zahl bilden
21040	16000	Einzelterm auswerten
21130	18000	Symbol in Tabelle eintragen

```
!=========================================================!
! 23040! 17000 ! Fehler registrieren                      !
! 24030!    50 ! Position von A$ in AA$ bestimmen         !
!=========================================================!
!                                                         !
! Verzweigungen nach außen :                              !
!                                                         !
!---------------------------------------------------------!
! in Ze ! nach  ! Bedingung          ! Bemerkung          !
!---------------------------------------------------------!
! 20060 ! STOP  ! DS NE 0            ! Disk-Fehler        !
! 20170 ! RETURN!                    ! Ende von Upro 20000!
! 21140 ! RETURN!                    ! Ende von Upro 21000!
! 22140 ! RETURN!                    ! Ende von Upro 22000!
! 23040 ! RETURN! B LT 0 OR B GT 1   ! Ungült. Binärziffer!
! 23070 ! RETURN!                    ! Ende von Upro 23000!
! 24060 ! RETURN!                    ! Ende von Upro 24000!
! 25030 ! RETURN!                    ! Ende von Upro 25000!
!=========================================================!
```

5.4 Hauptprogramm mit Vorspann

Vorspann

```
50030 DIMAC(5)
50040 DEFFNV(X)=X-48+7*(X>64)
50050 FORI=0TO5
50060 READAC(I)
50070 NEXT
50080 K9=53
50090 DIMK$(K9),KM(K9),KC$(K9,10)
50100 FORI=1TOK9
50110 READK$(I),KM(I)
50120 IFAC(KM(I))=0THEN50140
50130 FORJ=0TOAC(KM(I))-1:READKC$(I,J):NEXTJ
50140 NEXTI
50150 D9=3
50160 DIMD$(D9)
50170 FORI=1TOD9
50180 READD$(I)
50190 NEXT
50200 DIMBK$(255)
50210 FORI=0TO127
50220 BK$(I)=CHR$(I)
50230 NEXT
50240 FORI=128TO203
50250 READBK$(I)
50260 NEXT
50270 E9=20
```

```
50280 DIMER$(E9)
50290 FORI=1TOE9
50300 READER$(I)
50310 NEXT
50320 SP$="
50330 HE$="0123456789ABCDEF"
50340 AN$=CHR$(34)
50350 MT=300
50360 DIMTY$(MT),TN$(MT),TV(MT),UA(MT),UT%(MT),UN$(MT)
50370 TA=0
50380 U=0
50390 SZ$="#(),.@!+-*/$'"
50400 DIMA(LEN(SZ$))
50410 DIMEB(10)
50420 OPEN15,8,15,"I"
50430 GOSUB25000
50440 IFDSTHENPRINTDS$:STOP
50450 GOTO1030
```

In Zeile 1 des Programms steht ein Sprung auf Zeile 50000, denn aus Rechenzeitgründen ist es sinnvoll, den Vorspann eines Programms zeilennummernmäßig an den Schluß zu legen. In diesem Vorspann werden im wesentlichen die Felder mit den Werten aus den DATA-Statements besetzt sowie einige globale Variablen mit konstanten Werten belegt. Außerdem wird in Zeile 50040 eine Funktion FNV(X) definiert, die zur Berechnung von hexadezimalen Werten nützlich ist. In Zeile 50420 wird das Floppylaufwerk initialisiert und anschließend zur Zeile 1030 gesprungen, wo das eigentliche Hauptprogramm beginnt.

Assemblieren

```
1030 INPUT"DATEI                ";F$
1040 TY=1
1050 OPEN2,8,2,F$+",SRC,S"
1060 GOSUB25000
1070 IFDS=0THEN1150
1080 IFDS<>64THENPRINTDS$:CLOSE2:GOTO1030
1090 CLOSE2
1100 TY=2
1110 OPEN2,8,2,F$+",SRC,P"
1120 GOSUB25000
1130 IFDSTHENPRINTDS$:CLOSE2:GOTO1030
1140 GET#2,A$,A$:REM STARTADRESSE UEBERLESEN
1150 PRINT"■■EINGABEDATEI IST "F$",SRC"
1160 OPEN3,8,1,"@:"+F$+",,,,P,W"
1170 GOSUB25000
1180 IFDSTHENPRINTDS$:STOP
```

```
1190 INPUT"■LIST-GERAET (ADR.)   4■■■■";LD
1200 PRINT"■LISTE ZU ";
1210 IFLD>1THENOPEN4,LD:PRINT"GERAET";LD:GOTO1260
1220 PRINT"DATEI "F$".LST AUF GERAETENR."LD
1230 OPEN4,LD,4,"@:"+F$+".LST,S,W"
1240 GOSUB25000
1250 IFDSTHENPRINTDS$:STOP
1260 PRINT#4,"*** COMMODORE 64   6502-ASSEMBLER ***     ";
1270 PRINT#4,"     VERSION 1.1  (16.11.83)"
1280 PRINT#4,"ASSEMBLIEREN   VON "F$".SRC"
1290 PRINT#4,"OBJECT-DATEI   IST "F$".OBJ"
1300 PRINT#4,"SYMBOL-TABELLE IST "F$".SYM"
1310 PRINT#4
1320 PRINT#4,"ZEILE  ADR.  OBJ    * QUELLTEXT"
1330 PRINT#4
1340 GOSUB22000
1350 TT$=T$
1360 PRINTZN$,AD,TT$
1370 IFLD=3THENPRINT"□";
1380 GOSUB10000
1390 HH=AD
1400 GOSUB250
1410 PRINT#4,ZN$"  "HH$;"  "LEFT$(C$+"        ",7);U$" "TT$
1420 IFEBTHENFORI=1TOEB:PRINT#4,"FEHLER: ";ER$(EB(I)):NEXT
1430 EB=0
1440 IFC$=""THEN1510
1450 AD=AD+1
1460 H$=LEFT$(C$,2)
1470 GOSUB350
1480 PRINT#3,CHR$(H);
1490 C$=MID$(C$,3)
1500 GOTO1440
1510 IFIS=0THEN1340
1520 IFIS=64THENPRINT#4,"■DATEIENDE ERREICHT.
1530 PRINT#4
1540 PRINT#4,EN"FEHLER."
1550 IFLD<>3THENPRINTEN"FEHLER."
1560 CLOSE3
1570 CLOSE2
```

Die Zeilen 1030 bis 1570 enthalten den kompletten Ablauf zum Assemblieren einer Datei, ausschließlich dem Einsetzen der bisher undefinierten Symbole in die Objekt-Datei, was ab Zeile 2000 bewerkstelligt wird.

In Zeile 1030 wird zunächst der Name der zu übersetzenden Datei in die Variable F$ vom Bildschirm eingelesen. Der Dateiname wird hier ohne den Zusatz '.SRC' angegeben. Die Quelldatei kann entweder eine sequentielle Datei oder eine Programmdatei sein. Diese Datei benennt man, um Verwechslungen zu vermeiden, am besten mit dem Anhang '.SRC' für Source-Code. Die Art der Quelldatei wird vom Programm sel-

bstständig erkannt und in der Variablen TY festgehalten. Dabei bedeutet TY=1, daß die Eingabedatei sequentiell ist, TY=2 bedeutet, daß die Eingabedatei eine Programmdatei ist. Die Unterscheidung geschieht dadurch, daß zunächst versucht wird, die Eingabedatei als sequentielle Datei zu öffnen. War die geöffnete Datei aber eine Programmdatei, so erkennt das Floppy-Betriebssystem dies und gibt den Fehlercode 64 zurück. Daraufhin versucht das Programm die Datei als Programmdatei zu öffnen.

Eine Programmdatei enthält in den ersten beiden Bytes die Startadresse, wohin es geladen werden soll. Diese Startadresse ist jedoch hier unerheblich und muß deshalb überlesen werden, was in Zeile 1140 geschieht, wenn die Eingabedatei eine Programmdatei ist.

In Zeile 1160 wird als File #3 eine Zwischendatei eröffnet, mit dem Namen der Datei und angehängten vier Punkten. In diese Datei wird die vorläufige Version der Objektdatei geschrieben. In Zeile 1190 kann das Ausgabegerät angewählt werden. Möglich sind dabei die Eingaben 4 für Drucker, 3 für Bildschirm und 8 für Floppy. Wenn als Ausgabegerät die Floppy spezifiert wurde, wird eine sequentielle Datei angelegt, auf der genau das abgespeichert wird, was sonst auf dem Drucker erscheint.

Die Zeilen 1260 bis 1330 bilden die Kopfzeilen des Listings. Anschließend werden in einer Schleife jeweils einzeln die Zeilen aus dem Source-Text eingelesen und in der Variablen TT$ gespeichert. Diese Zeilen werden dann sogleich assembliert und das Ergebnis auf der Listdatei ausgegeben. Dann werden noch die Fehlermeldungen ausgegeben, falls welche vorhanden waren. Die Anzahl der Fehlermeldungen pro Zeile wird sogleich auf 0 zurückgesetzt. In der Variablen C$ ist nach dem Assemblieren eine hexadezimale Befehlsfolge gespeichert. Diese wird dann als Gruppe von zwei Zeichen, die jeweils ein Byte bilden, in die vorläufige Objekt-Datei gechrieben, bis die Variable C$ die Länge 0 hat. In Zeile 1510 wird zum Einlesen der nächste Zeile gesprungen, wenn IS gleich 0 ist, d.h. wenn das Ende der Datei noch nicht erreicht ist.

War die Datei zu Ende, so wird eine entsprechende Meldung ausgegeben, sowie die Anzahl der Fehler auf der Listdatei und am Bildschirm ausgegeben und sowohl die Eingabedatei als auch die vorläufige Objektdatei geschlossen. Damit ist das Programmstück 'Assemblieren' beendet.

Werte nachträglich einsetzen

```
2010 PRINT#15,"S:"+F$+".OBJ"
2020 GOSUB25000
2030 IFDS>1THENPRINTDS$:STOP
2040 IFUTHEN2090
2050 PRINT#15,"R:"+F$+".OBJ="+F$+"...."
2060 GOSUB25000
2070 IFDS<>1THENPRINTDS$:STOP
2080 GOTO2440
2090 OPEN3,8,1,F$+".OBJ,P,W"
2100 GOSUB25000
2110 IFDS<>1THENPRINTDS$:STOP
2120 OPEN2,8,2,F$+"....,P,R"
2130 GOSUB25000
2140 IFDS<>1THENPRINTDS$:STOP
2150 AD=SA-3:REM STARTADRESSE UEBERLESEN
2160 U1=1
2170 AD=AD+1
2180 GOSUB10
2190 IFUA<U1>>ADORU1>UTHENPRINT#3,CHR$(II);:GOTO2390
2200 UT=UT%(U1)
2210 SI=0
2220 IFUT=1ORUT=3ORUT=4ORUT=8THENSI=1
2230 IFUT=2ORUT=5ORUT=6ORUT=7THENSI=2
2240 IFSI=2THENI1=II:GOSUB10:I2=II
2250 IFUT=0THENSTOP
2260 TN$=UN$(U1)
2270 GOSUB500
2275 IFTF=0THENGOSUB21000
2280 IFUT=8THEN2350
2290 IFSI=1THENPRINT#3,CHR$(II-128+TV);:GOTO2380
2300 A=I1+256*I2-32768-128+TV
2310 H=INT(A/256)
2320 PRINT#3,CHR$(A-256*H)CHR$(H);
2330 AD=AD+1
2340 GOTO2380
2350 A=II+256*(II>127)+TV-(AD+1)
2360 IFA<0THENA=A+256
2370 PRINT#3,CHR$(A);
2380 U1=U1+1
2390 IFIS=0THEN2170
2400 CLOSE2
2410 PRINT#15,"S:"+F$+"...."
2420 GOSUB25000
2430 IFDS>1THENPRINTDS$:STOP
2440 PRINT#4,"FERTIG ASSEMBLIERT."
2450 IFLDC<>3THENPRINT"FERTIG ASSEMBLIERT."
2460 GOSUB20000
2470 CLOSE15
```

```
2480 CLOSE4
2490 END
```

In diesem Programmstück von Zeile 2010 bis 2490 wird aus der vorläufigen Objektdatei die endgültige Fassung gebildet, indem alle bisher undefinierten Symbole jeweils an die richtige Stelle eingetragen werden.

Dazu wird zunächst eine eventuell schon vorhandene Version der Objektdatei gelöscht. War kein Symbol undefiniert (U=0) wird einfach die vorläufige Objektdatei zur endgültigen Objektdatei umbenannt, was in Zeile 2050 geschieht. Ansonsten wird eine endgültige Objektdatei als File #3 eröffnet und die bisherige Objektdatei als File #2.

Es wird jeweils ein Zeichen aus File #2 gelesen und wieder in File #3 weggeschrieben, solange bis die in der Variablen AD mitgezählte Adresse gleich der Adresse des ersten undefinierten Symbols ist. Dann wird anstatt dem Wert in der vorläufigen Quelldatei der richtige Wert in die endgültige Objektdatei geschrieben. Die Berechnung des endgültigen Wertes ist jedoch nicht ganz einfach. Insbesondere muß festgestellt werden, ob ein oder zwei Byte korrigiert werden müssen. Die Anzahl der zu korrigierenden Bytes wird in der Variablen SI festgehalten. Wenn der Typ (UT) des undefinierten Symbols gleich 1, 3, 4 oder 8 ist, muß ein Byte ersetzt werden, ansonsten zwei Byte.

In den Zeilen 2260 bis Zeilen 2275 wird der Wert des undefinierten Symbols festgestellt. Wenn der Name nicht in der Tabelle enthalten ist (TF=0) so wird der Wert mit Hilfe des Unterprogramm ab Zeile 21000 vom Bildschirm eingelesen.

Die Berechnung der einzusetzenden Werte muß unterschiedlich erfolgen, je nachdem, ob das undefinierte Symbol aus einem relativen Sprung resultiert (UT=8), oder ob ein oder zwei Byte korrigiert werden müssen.

Die Behandlung der relativen Sprünge geschieht in den Zeilen 2350 bis 2370, ein Byte wird in Zeile 2290 korrigiert und zwei Bytes in den Zeilen 2300 bis 2330.

Bei Zeile 2380 vereinigen sich die drei Zweige des Programms wieder. Dort wird die Variable U1, die auf das nächste zu ersetzende Symbol zeigt, um eins erhöht. Wenn die Datei noch nicht zu Ende ist (IS=0), wird zur Zeile 2170 gesprungen, wo das nächste Byte der vorläufigen Objektdatei gelesen wird. Wurde das Ende der Datei erreicht, so wird die Zwischendatei gelöscht und die Meldung 'Assem-

blierung fertig' ausgegeben. Danach wird jedoch noch die Symboltabelle auf Drucker und auf Floppy mit Hilfe des Unterprogramms auf Zeile 20000 ausgegeben.

Der Assembler hat also im Endeffekt zwei Dateien neu angelegt. Zum einen die endgültige Objektdatei und zum anderen eine sequentielle Datei, die die Symbole enthält. Diese Symboldatei kann mit erweiterten Assembler-Versionen zum Verbinden von mehreren Programmen benützt werden.

Die Quelldatei wurde nur gelesen und deshalb nicht verändert.

Hauptprogramm mit Vorspann

Sprung auf Zeile 50000	1
Vorspann	50000 - 50450
Assemblieren	1030 - 1570
Werte nachträglich einsetzen	2000 - 2490

Variablen:

Name	Typ	Bereich	Bedeutung
A	H	-65535...65535	Hilfsvariable
A$	H	1 Zeichen	Eingelesenes Zeichen
AD	G	0...65535	Aktuelle Obj.-Adresse
AN$	G	1 Zeichen	Anführungszeichen
C$	R	0..6 Zeichen	Assemblierter Code
D9	G	Konstante 3	Anz. Direktiven
DS	R	0..99	Disk-Status (Nummer)
DS$	R	Zeichenreihe	DS,D1$,D2$,D3$ kombin.
E9	G	Konstante 20	Anz. Fehlermeldungen
EB	R	0...10	Anzahl Fehler in Zeile
EN	G	Ganzzahlig	Anzahl Fehler insges.
F$	G	Zeichenreihe	Dateiname
H	R/H	0...65535	Wert v. 2-stell. Hex-Z
H$	P	2 Zeichen	2-stellige Hex-Zahl
HE$	G	01234567890abcedf	Hexadezimal-Ziffern
HH	P	0...65535	Wert v. 4-stell. Hex-Z
HH$	R	4 Zeichen	4-stellige Hex-Zahl
I	H	0...300	Laufvariable

I1	H	0...255	1. Eingelesener Code
I2	H	0...255	2. Eingelesener Code
II	R	0...255	Eingelesener Code
IS	R	0...320	Einlese-Status
J	H	0...10	Laufvariable
K9	G	Konstante 58	Anz. Mnemonics
LD	G	0..15	Ausgabe-Geräte-Adresse
MT	G	Konstante 300	Max. Tabellengröße
SA	H	0...65535	absolute Startadresse
SI	H	0...2	Anz. Byte im Operand
SP$	G	ca. 50 Blanks	z. Auffüll. v. Strings
SZ$	G	"#(),.%!+-*/$'"	Alle Sonderzeichen
T$	R/P	Zeichenreihe	Eingelesene Zeile
TT$	H	Zeichenreihe	Kopie von T$
TA	G	0...300	Anzahl Symbole in Tab.
TF	R	0...300	Nummer des Symbols
TN$	E	Zeichenreihe	Gesuchtes Symbol
TV	E/A	0..65535	Wert des Symbols
TY	P	1 oder 2	Typ der Quell-Datei
TY$	E/A	Zeichenreihe	Typ des Symbols
U	G	0...300	Anz. undef. Symbole
U1	H	0...300	zeigt auf nächstes undefinierte Symbol
UN$	E	Zeichenreihe	Name des und. Symbols
UT	E	0...8	Typ des undef. Symbols
W	A	0...65535	Wert des Einzelterms
ZN$	R	5 Zeichen	Zeilennr. als String

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung
A	LEN(SZ$)	G	0...255	Positionen der Sonderzeichen
AC	5	H	0...10	Anzahl Modes je Befehlstyp
BK$	203	G	Zeichenreihen	Basic-Befehle
D$	D9	G	Zeichenreihen	Directiven
EB	10	G	0...E9	Fehlernummern
ER$	E9	G	Zeichenreihen	Fehlermeldungen
K$	K9	G	je 3 Zeichen	Mnemonics
KM	K9	G	0...5	Befehlstyp
KC$	K9,10	G	2 Hex-Ziffern	Maschinencodes
TN$	300	G	Zeichenreihen	Symbole
TV	300	G	0...65535	Werte d. Symbole
TY$	300	G	4 Zeichen	Typen d. Symbole
UA	300	G	0...65535	Adressen der

UN$	300	G	Zeichenreihen	undef. Symbole Namen der undef. Symbole
UT%	300	G	0...8	Typ d. und. Sym.

Dateien :

#	Name	T	Bemerkung
2	F$+".SRC"	s/p	Quelldatei
3	F$+"...."	p	Vorläufige Objektdatei
4	F$+".LST"	s	Listdatei
3	F$+".OBJ"	p	Endgültige Objektdatei
15	Floppy		Kommado-/Fehlerkanal der Floppy

Unterprogrammaufrufe :

in	nach	Zweck
1060	25000	Disk-Status abfragen
1120	25000	Disk-Status abfragen
1240	25000	Disk-Status abfragen
1340	22000	Eine Zeile einlesen
1380	10000	Eine Zeile assemblieren
1400	250	4-stellige Hex-Zahl bilden
1470	350	Wert von 2-stelliger Hex-Zahl bestimmen
2020	25000	Disk-Status abfragen
2060	25000	Disk-Status abfragen
2100	25000	Disk-Status abfragen
2130	25000	Disk-Status abfragen
2180	10	Ein Zeichen aus Zwischendatei einlesen
2240	10	Ein Zeichen aus Zwischendatei einlesen
2270	500	Symbol in Tabelle suchen
2275	21000	Symbol manuell erfassen
2420	25000	Disk-Status abfragen
2460	20000	Symboltabelle ausgeben
50430	25000	Disk-Status abfragen

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
1180	STOP	DS NE 0	Disk-Fehler

1250	STOP	DS NE 0	Disk-Fehler
2030	STOP	DS GT 1	Disk-Fehler
2110	STOP	DS NE 0	Disk-Fehler
2140	STOP	DS NE 0	Disk-Fehler
2250	STOP	UT = 0	darf eigentlich nicht auftauchen
2430	STOP	DS GT 1	Disk-Fehler
2490	END		Normales Ende

5.5 Bedienungsanleitung

In diesem Kapitel wird beschrieben, wie der Assembler grundsätzlich handzuhaben ist.

Die Erstellung der Quell-Datei kann mit einem Texteditor oder einfacher mit dem normalen Basic-Editor vorgenommen werden. Anschließend speichert man das editierte Programm mit dem Befehl SAVE"'Name'.SRC",8 auf Diskette ab. Das Wort 'Name' kann dabei durch einen selbst gewählten Dateinamen ersetzt werden.

Anschliessend wird der Assembler mit der Befehlsfolge

```
LOAD "ASS64",8
RUN
```

gestartet. Der Assembler fragt dann nach dem Namen der Quell-Datei (Source). Hier ist also der Name der Datei (ohne den Zusatz '.SRC') einzugeben.

Für das Ausgabegerät für die Liste hat man folgende Eingabemöglichkeiten:

3 - Ausgabe auf Bildschirm
4 - Ausgabe auf Drucker
8 - Ausgabe auf eine sequentielle Datei "'Name'.LST"

Das Programm wird nun assembliert. Vom Assembler erkannte Fehler im Quelltext werden auf der Liste mitprotokolliert.

Wurde ein Symbol oder ein Label nicht definiert, so wird keine Fehlermeldung ausgegeben, sondern der Wert des Symbols am Bildschirm erfragt.

Nachdem fertig assembliert wurde, wird die Symboltabelle noch an die Liste angefügt und auf Floppy gespeichert, wo

man sie unter dem Namen "'Name'.SYM" lesen kann.

Das erzeugte Objekt-Programm kann mit dem Befehl

```
LOAD "'Name'.OBJ",8,1
```

an die im Quell-Code angegebene Startadresse geladen werden. Die Angabe der '1' als Sekundäradresse ist notwendig, weil sonst die gespeicherte Startadresse von der Basic-Laderoutine ignoriert wird.

Will man die Objekt-Datei von einem Basic-Programm aus laden, so geschieht dies in folgender Weise:

```
10 IF L=0 THEN L=1 : LOAD "'Name'.OBJ",8,1
20 REM PROGRAMMBEGINN
```

Nach der Ausführung eines LOAD-Befehls startet das Programm immer wieder in der ersten Zeile. Man muß sich also einen Merker setzen, daß der LOAD-Befehl ausgeführt wurde.

6

BASIC-Erweiterungen und Änderungen

6. Basic-Erweiterungen und Änderungen

Das folgende Kapitel soll dazu dienen, Sie in die Lage zu versetzen, Basic-Erweiterungen selbstständig durchzuführen, bzw. Ihr Basic den eigenen Wünschen anzupassen.

Hilfsmittel dazu ist natürlich der in Kapitel 5 vorgestellte Assembler, da Basic-Erweiterungen natürlich in Maschinensprache formuliert sein müssen. Im ersten Teil wollen wir die generelle Vorgehensweise beschreiben, wie eigene Befehle in die vorhandene Befehls-Tabelle eingebaut werden können. Im zweiten Teil wollen wir die in Kapitel 4.2 beschriebenen Grafikerweiterungen mit Hilfe des in Kapitel 5 beschriebenen Assemblers um einiges verschnellern.

Als letztes werden wir noch einen allgemeinen Basic-Befehl (PAUSE) vorstellen, sowie eine Änderung des bestehenden Basic und natürlich die Routine zum Abschalten der Basic-Erweiterung.

6.1 Vorgehenweise bei Basic-Erweiterungen

Prinzipiell gibt es dazu zwei Vorgehensweisen. Entweder man kopiert den kompletten Basic-Interpreter vom ROM in das RAM und ändert dann diese Kopie nach seinen eigenen Wünschen ab, oder man verstellt gewisse Basic-Vektoren auf seine eigenen Routinen.

Da die erste Art in Kapitel 6.4 beschrieben wird, wird hier im folgenden eine Übersicht über die Vektoren des Basic-Interpreters gegeben, die leicht verstellbar sind, weil sie von Hause aus im RAM stehen.

Adresse des Vektors	normaler Wert	Bemerkung
$0300	$E38B	Fehlermeldung ausgeben
$0302	$A483	Basic-Warmstart
$0304	$A57C	Umwandlung einer Basic-Zeile in Interpreter-Code
$0306	$A71A	Umwandlung von Basic-Code in Klartext
$0308	$A7E4	Basic-Befehl ausführen
$030A	$AE86	numerischen Ausdruck auswerten

Am wichtigsten für Basic-Erweiterungen ist der vorletzte Vektor. Man stellt ihn zweckmäßigerweise auf eine Routine, die überprüft, ob das eingelesene Zeichen ein Basic-Befehl oder ein eigener Befehl ist.

Das Assembler-Listing für ein mögliches Programm zum Dekodieren eigener Basic-Befehle ist im folgenden wiedergegeben.

```
*** COMMODORE 64   6502-ASSEMBLER ***        VERSION 1.1
ASSEMBLIEREN   VON BASERW.SRC
OBJECT-DATEI   IST BASERW.OBJ
SYMBOL-TABELLE IST BASERW.SYM

ZEILE  ADR.  OBJ     * QUELLTEXT

   10  C800           *ORG $C800
   20  C800           ;*******************************
   30  C800           ;*  STARTADRESSE = 51200     *
   40  C800           ;*******************************
   50  C800           ;
   60  C800           ;
   70  C800           ;
  100  C800           ;*******************************
  110  C800           ;* KONSTANTENVEREINBARUNGEN *
  120  C800           ;*******************************
  130  C800           DECODE=$0308  ; ADRESSE DES VEKTORS
  131  C800           ;DEKODIER-ROUTINE
  132  C800           ;
  150  C800           BASBZ=$7A     ; ENTHAELT BASIC-
  151  C800           ;BEFEHLS-ZEIGER
  152  C800           ;
  500  C800           ;*******************************
  510  C800           ;* BASIC-ROUTINEN              *
  520  C800           ;*******************************
  525  C800           ;
  530  C800           CHRGET=$0073;EIN BASIC-ZEICH. LESEN
  531  C800           ;
  540  C800           INTERPRET=$A7AE;INTERPRETERSCHLEIFE
  541  C800           ;
  550  C800           BASICBEF=$A7E7 ;ARBEITET BASIC-
  551  C800           ;               BEFEHL AB
  552  C800           ;
```

```
1000  C800             ;*******************************
1010  C800             ;*  INITIALISIERUNG            *
1020  C800             ;*******************************
1030  C800             INIT:
1040  C800  A980    R  LDA #BASERWL
1050  C802  8D0803    STA DECODE
1060  C805  A980    R  LDA #BASERWH
1070  C807  8D0903    STA DECODE+1 ; DECODE-VEKTOR
1075  C80A             ;AUF NEUE BEFEHLE STELLEN
1080  C80A  60         RTS ; INITIALISIERUNG FERTIG
1090  C80B             ;
1200  C80B             ;*******************************
1210  C80B             ;*  ERWEITERUNG AUSSCHALTEN    *
1220  C80B             ;*******************************
1230  C80B             AUS:
1240  C80B  A9E4       LDA #$E4
1250  C80D  8D0803     STA DECODE
1260  C810  A9A7       LDA #$A7
1270  C812  8D0903     STA DECODE+1 ; DECODE-VEKTOR
1275  C815             ;AUF NORMALEN WERT STELLEN
1280  C815  60         RTS ; FERTIG
1290  C816             ;
1500  C816             ;*******************************
1510  C816             ;* HILFSZELLEN                 *
1520  C816             ;*******************************
1530  C816  00         BEFNR: BYT 0 ; NUMMER DES BEFEHLS
1531  C817             ;
1540  C817  00         AKKUSICH: BYT 0 ;AKKU SICHERN
1541  C818             ;
2000  C818             ;*******************************
2010  C818             ;* NEUE BASIC-BEFEHLE AUSF.    *
2020  C818             ;*******************************
2030  C818             BASERW:
2040  C818             BASERWH = $ / 256
2050  C818             HILF = BASERWH *256
2060  C818             BASERWL = BASERW - HILF
2070  C818             ; BASERWH,L BERECHNET
2080  C818  207300     JSR CHRGET ; BASIC-ZEICHEN HOLEN
2090  C81B  9000    R  BCC BEBAS ;BASIC-CODE WAR Z'IFFER
2100  C81D  C960       CMP #$60 ; TEST AUF BUCHSTABE
2110  C81F  B000    R  BCS BEBAS; BASIC-BEFEHLS-CODE
2120  C821  C941       CMP #$41 ; TEST AUF BUCHSTABE 'A'
2130  C823  9000    R  BCC BEBASC; SONDERZEICHEN
2140  C825  8D17C8     STA AKKUSICH ; AKKU SICHERN
2150  C828  A200       LDX #0
2160  C82A  8E16C8     STX BEFNR ; BEFNR =0
```

```
2170  C82D             BE1:
2180  C82D  A000       LDY #0
2190  C82F  EE16C8     INC BEFNR ;BEFNR=BEFNR+1
2200  C832  BD8080 R   LDA BTAB,X;ZEICHEN AUS BEF.TABELLE
2210  C835  D000   R   BNE BE2   ;KEIN TRENNZEICHEN (0)
2220  C837  AD17C8     LDA AKKUSICH ;AKKU WIEDERHERSTELLEN
2230  C83A             BEBASC:
2240  C83A  38         SEC ; CARRY WIEDER SETZEN
2250  C83B             BEBAS:
2260  C83B  4CE7A7     JMP BASICBEF ;BASIC BEF.AUSFUEHREN
2270  C83E             BE2:
2280  C83E  D17A       CMP (BASBZ),Y;VERGL. M. BASIC-TEXT
2290  C840  D000   R   BNE BE4 ; KEINE UEBEREINSTIMMUNG
2300  C842  C8         INY ;NAECHST. ZEICH. IM BASIC-TEXT
2310  C843  E8         INX ;NAECHST. ZEICH. IN BEF.TAB.
2320  C844  BD8080 R   LDA BTAB,X ;ZEICHEN AUS BEF.TAB.
2330  C847  D0F5       BNE BE2 ; NAECHSTES ZEICH. PRUEFEN
2340  C849  18         CLC
2350  C84A  98         TYA
2360  C84B  657A       ADC BASBZ ;BEFEHLSZEIGER UM
2370  C84D  857A       STA BASBZ ;BEFEHLSLAENGE ERHOEHEN
2380  C84F  9000   R   BCC BE3   ;KEIN UEBERTRAG
2390  C851  E67B       INC BASBZ+1 ;UEBERTRAG
2400  C853             BE3:
2410  C853  AD16C8     LDA BEFNR; BEFEHLSNUMMER HOLEN
2420  C856  0A         ASL A     ; VERDOPPELN
2430  C857  AA         TAX       ;ALS ZEIGER IN SPRUNGTAB.
2440  C858  BD8080 R   LDA STAB,X
2450  C85B  8D8080 R   STA BEFADR
2460  C85E  BD8180 R   LDA STAB+1,X
2470  C861  8D8180 R   STA BEFADR+1  ; BEFADR=SPRUNGZIEL
2480  C864             ;
2490  C864  20         BYT $20       ; CODE FUER 'JSR'
2500  C865  0000       BEFADR: WOR 0 ; SPRUNGZIEL
2510  C867  4CAEA7     JMP INTERPRET ; ZURUECK ZUR
2520  C86A             ; INTERPRETER-SCHLEIFE
2530  C86A             ;
2540  C86A             BE4:
2550  C86A  E8         INX ;NAECHST. ZEICHEN IN BEF.TAB.
2560  C86B  BD8080 R   LDA BTAB,X
2570  C86E  D0FA       BNE BE4     ;BEFEHL NICHT ZU ENDE
2580  C870  E8         INX         ;'0' UEBERLESEN
2590  C871  4C2DC8     JMP BE1     ;NAECHST. BEF. CHECKEN
2600  C874             ;
```

```
3000  C874           ;*********************************
3010  C874           ;*  SPRUNGTABELLE                *
3020  C874           ;*********************************
3030  C874           STAB:
3050  C874  E7A7     WOR BASICBEF  ;BEFEHR=A
3060  C876  00C0     WOR $C000     ;PUNKT
3070  C878  03C0     WOR $C003     ;GREIN
3080  C87A  0CC0     WOR $C00C     ;GRAUS
3090  C87C  0FC0     WOR $C00F     ;LINIE
3100  C87E  12C0     WOR $C012     ;RECHT
3110  C880  15C0     WOR $C015     ;BLOCK
3120  C882  18C0     WOR $C018     ;KREIS
3130  C884  1BC0     WOR $C01B     ;RADIUS
3140  C886  00CA     WOR $CA00     ;PAUSE
3150  C888  0BC8     WOR AUS       ;AUS
3160  C88A           ;
4000  C88A           ;*********************************
4010  C88A           ;*  BEFEHLSTABELLE               *
4020  C88A           ;*********************************
4030  C88A           BTAB:
4040  C88A  50       BYT $50 ;'P'
4050  C88B  55       BYT $55 ;'U'
4060  C88C  4E       BYT $4E ;'N'
4070  C88D  4B       BYT $4B ;'K'
4080  C88E  54       BYT $54 ;'T'
4090  C88F  00       BYT 0 ;TRENNZEICHEN
4100  C890  47       BYT $47 ;'G'
4110  C891  52       BYT $52 ;'R'
4120  C892  45       BYT $45 ;'E'
4130  C893  49       BYT $49 ;'I'
4140  C894  4E       BYT $4E ;'N'
4150  C895  00       BYT 0 ;TRENNZEICHEN
4160  C896  47       BYT $47 ;'G'
4170  C897  52       BYT $52 ;'R'
4180  C898  41       BYT $41 ;'A'
4190  C899  55       BYT $55 ;'U'
4200  C89A  53       BYT $53 ;'S'
4210  C89B  00       BYT 0 ;TRENNZEICHEN
4220  C89C  4C       BYT $4C ;'L'
4230  C89D  49       BYT $49 ;'I'
4240  C89E  4E       BYT $4E ;'N'
4250  C89F  49       BYT $49 ;'I'
4260  C8A0  45       BYT $45 ;'E'
4270  C8A1  00       BYT 0 ;TRENNZEICHEN
4280  C8A2  52       BYT $52 ;'R'
4290  C8A3  45       BYT $45 ;'E'
4300  C8A4  43       BYT $43 ;'C'
4310  C8A5  48       BYT $48 ;'H'
4320  C8A6  54       BYT $54 ;'T'
4330  C8A7  00       BYT 0 ;TRENNZEICHEN
```

```
 4340  C8A8  42        BYT $42 ;'B'
 4350  C8A9  4C        BYT $4C ;'L'
 4360  C8AA  4F        BYT $4F ;'O'
 4370  C8AB  43        BYT $43 ;'C'
 4380  C8AC  4B        BYT $4B ;'K'
 4390  C8AD  00        BYT 0 ;TRENNZEICHEN
 4400  C8AE  4B        BYT $4B ;'K'
 4410  C8AF  52        BYT $52 ;'R'
 4420  C8B0  45        BYT $45 ;'E'
 4430  C8B1  49        BYT $49 ;'I'
 4440  C8B2  53        BYT $53 ;'S'
 4450  C8B3  00        BYT 0 ;TRENNZEICHEN
 4460  C8B4  52        BYT $52 ;'R'
 4470  C8B5  41        BYT $41 ;'A'
 4480  C8B6  44        BYT $44 ;'D'
 4490  C8B7  49        BYT $49 ;'I'
 4500  C8B8  55        BYT $55 ;'U'
 4510  C8B9  53        BYT $53 ;'S'
 4520  C8BA  00        BYT 0 ;TRENNZEICHEN
 4530  C8BB  50        BYT $50 ;'P'
 4540  C8BC  41        BYT $41 ;'A'
 4550  C8BD  55        BYT $55 ;'U'
 4560  C8BE  53        BYT $53 ;'S'
 4570  C8BF  45        BYT $45 ;'E'
 4580  C8C0  00        BYT 0 ;TRENNZEICHEN
 4590  C8C1  41        BYT $41 ;'A'
 4600  C8C2  55        BYT $55 ;'U'
 4610  C8C3  53        BYT $53 ;'S'
 4620  C8C4  00        BYT 0 ;TRENNZEICHEN
 4990  C8C5  00        BYT 0 ;ENDE BEFEHLSTABELLE
59990  C8C6            ;*******************************
59991  C8C6            ;*                             *
59992  C8C6            ;* PROGRAMMENDE                *
59993  C8C6            ;*                             *
59994  C8C6            ;*******************************
59999  C8C6            %END

 0 FEHLER.
ASSEMBLIERUNG FERTIG.
```

```
SYMBOLE:
NAME        TYP    WERT

DECODE      CONW   0308
BASBZ       CONW   007A
CHRGET      CONW   0073
INTERPRE    CONW   A7AE
BASICBEF    CONW   A7E7
INIT        LABL   C800
AUS         LABL   C80B
BEFNR       LABL   C816
AKKUSICH    LABL   C817
BASERW      LABL   C818
BASERWH     CONW   00C8
HILF        CONW   C800
BASERWL     CONW   0018
BE1         LABL   C82D
BEBASC      LABL   C83A
BEBAS       LABL   C83B
BE2         LABL   C83E
BE3         LABL   C853
BEFADR      LABL   C865
BE4         LABL   C86A
STAB        LABL   C874
BTAB        LABL   C88A
```

In diesem Listing sind drei kleine Programme zusammengefaßt. Das erste Programm 'INIT' setzt den Vektor der Befehlsdekodierung auf das Label BASERW. Wie die neuen Befehle eingeschaltet werden, wird in Kapitel 6.5 erläutert.

Das zweite Programm stellt den neuen Basic-Befehl 'AUS' dar, der die Basic-Erweiterung wieder abschaltet, indem der Vektor wieder mit dem alten Wert belegt wird.

Das Kernstück des Programms ist aber das Programm BASERW, das wir uns im folgenden nun genauer betrachten wollen.

Die normale Basic-Befehl-Dekodierroutine beginnt damit, daß ein Zeichen des Basic-Textes mit Hilfe der Routine CHRGET eingelesen wird. Wir wollen deshalb unsere eigene Dekodier-Routine genauso beginnen lassen. Wir brauchen das Zeichen, das eingelesen wurde, nur dann zu dekodieren, wenn es ein Buchstabe war. Die Routine CHRGET setzt das Carry-Flag, wenn das eingelesene Zeichen keine Ziffer war. Wir können also unsere Routine sofort überspringen, wenn das Carry-Flag 0 ist, ebenso wenn das eingelesene Zeichen größer als der ASCII-Code des letztmöglichen Buchstabens oder kleiner als der ASCII-Code vom Anfangszeichen 'A' ist. Zu beachten ist, daß wir, wenn keiner unserer Befehle gefunden wurde, nicht auf die Adresse springen dürfen, auf die der Dekodiervektor normalerweise zeigt, sondern einen Befehl weiter, also auf $A7E7, weil wir ja bereits ein Zeichen eingelesen haben.

In den Zellen $7A und $7B befindet sich der Basic-Befehlszähler. Durch die Kenntnis dieser Adresse brauchen wir nicht die folgenden Zeichen des Basic-Textes einzulesen, sondern können mit Hilfe eines indizierten Vergleichs jeweils ein Wort der Befehlstabelle mit dem Wort im Basictext vergleichen. Die Befehlstabelle ist so aufgebaut, daß jeweils ein Wort als Bytereihe angegeben wird, und anschließend eine 0 als Trennzeichen folgt. Das Ende der Befehlstabelle ist durch zwei aufeinanderfolgende Nullen gekennzeichnet. Jedesmal, wenn ein Befehl nicht gefunden wurde, wird die Nummer des auszuführenden Befehls, die in der Hilfszeile BEFNR gespeichert ist, um eins erhöht. Mit dem so gewonnen Wert wird anschließend mit Hilfe der Sprungtabelle die effektive Befehlsadresse ermittelt. In der Sprungtabelle stehen nacheinander als Wort die Sprungadressen für jeden Befehl, beginnend mit der Befehlsnummer 0, die allerdings keine Bedeutung hat. Die Befehlsadresse erhält man dadurch, daß man die Nummer des Befehls verdoppelt und als Zeiger in die Sprungtabelle verwendet. Lower- und Higher-Byte der Sprungadresse werden in zwei Speicherzellen geschrieben, die im Anschluß hinter dem hexadezimalen Code $20 folgen. Dieser Code bedeutet JSR, also einen Unterprogrammaufruf, und zwar wird gerade unser Unterprogramm aufgerufen, das wir mit Hilfe der Befehlsnummer ausgewählt haben. Nach Aufruf unseres Unterprogrammes wird wieder zur Interpreterschleife verzweigt.

Der Ablauf des Programms konnte hier aus Platzgründen nur kurz skizziert werden, doch werden wir in Band 3 genauer auf die verschiedenen Arten der Basic-Erweiterungen eingehen.

Wenn man selbst in dieses Programm noch einen weiteren Befehl einfügen möchte, so muß man einfach den Befehl als Byte-Reihe an die Befehlstabelle anfügen, und die effektive Sprungadresse an die Sprungtabelle anfügen. Mit diesem Algorithmus können bis zu 127 neuen Basic-Befehle eingefügt werden.

Die oben erwähnten anderen Vektoren können dazu benutzt werden, eigene Funktionen zu definieren oder eigene Fehlermeldungen auszugeben oder ähnliche Basic-Erweiterungen. Darauf kann hier allerdings nicht eingegangen werden (vgl. dazu Band 3).

6.2 Grafik-Erweiterungen

In diesem Kapitel wird sukzessive ein Assembler-Listing besprochen, mit dessen Hilfe die Befehle: PUNKT, GRafik EIN, GRafik AUS, LINIE, RECHT, BLOCK, KREIS und RADIUS realisiert werden können. Die Programmierung weiterer Befehle wie Segment oder Quader bzw. Quader voll, sowie weitere geometrische Figuren, können dem interessierten Leser als Übungsstoff dienen.

6.2.1 Konstanten-Vereinbarungen und ROM-Routinen

```
*** COMMODORE 64   6502-ASSEMBLER ***          VERSION 1.1
ASSEMBLIEREN   VON GRASS.SRC
OBJECT-DATEI   IST GRASS.OBJ
SYMBOL-TABELLE IST GRASS.SYM

ZEILE  ADR.  OBJ     * QUELLTEXT

   10  C000            .ORG $C000
   20  C000            ;*********************************
   30  C000            ;*  STARTADRESSE = 49152         *
   40  C000            ;*********************************
   50  C000            ;
   60  C000            ;
   70  C000            ;
  100  C000            ;*********************************
  110  C000            ;* KONSTANTENVEREINBARUNGEN      *
  120  C000            ;*********************************
  130  C000            ADRL=$14      ; INTEGERZAHL LOW-BYTE
  140  C000            ADRH=ADRL+1   ;       ''     HIGH-BYTE
  141  C000            ;ADRL,ADRH WERDEN VON DER ROUTINE
  142  C000            ;GETADR BZW. GETAB BESETZT
  143  C000            ;AUCH EINGABEPARAMETER (X-KOORD.)
```

```
144  C000            ;FUER 'PUNKT'
145  C000            ;
150  C000            PRDL=$57    ; PRODUKT BZW. DIVIDEND
160  C000            PRDH=PRDL+1
161  C000            ;PRDL,PRDH WERDEN VON DER ROUTINE
162  C000            ;MULT MIT DEM PRODUKT AUS FAK1
163  C000            ;UND FAK2 BESETZT.
164  C000            ;AUCH DIVIDEND FUER 'DIV'.
165  C000            ;
170  C000            FAK1L=$59     ; 1.FAKTOR BZW. DIVISOR
180  C000            FAK1H=FAK1L+1
181  C000            ;
190  C000            FAK2=$5B      ; 2.FAKTOR
191  C000            ;
200  C000            QUOTL=$5C     ; QOUTIENT
210  C000            QUOTH=QUOTL+1
211  C000            ;QUOTL,QUOTH WERDEN VON DER ROUTINE
212  C000            ;DIV MIT DEM QOUTIENTEN AUS PRD
213  C000            ;UND FAK1 BESETZT.
214  C000            ;
220  C000            Z=$FD         ; ZEIGER (HILFSVAR.)
221  C000            ;LAUFVARIABLE FUER BESCHREIBEN
222  C000            ;VON SPEICHERBERICHEN
223  C000            ;
230  C000            GANFH=$20 ;HIGH-BY ANF. GRAPHIK-RAM
231  C000            ;ANFANG GRAPHIK-RAM BEI $2000
232  C000            ;
240  C000            GENDH=$40 ;HIGH-BY ENDE GRAPHIK-RAM
241  C000            ;
250  C000            VANFANG=$0400 ;ANFANG VIDEO-RAM
251  C000            ;ANFANG VIDEO-RAM BEI $0400
252  C000            ;
260  C000            VIC=$D000 ;BASISADR. VIDEO-CONTR.
261  C000            ;
270  C000            INTL=$65 ; LOW-BYTE INT(FAC)
280  C000            INTH=$64 ; HIGH-BYTE INT(FAC)
281  C000            ;NACH AUFRUF VON 'FLPINT' STEHT IN
282  C000            ;INTL,INTH DER GANZZAHLIGE WERT DES
283  C000            ;FAC (FLIESSKOMMA-AKKUMULATORS)
290  C000            ;
291  C000            ;ES FOLGEN ADRESSEN VON FLIESSKOMMA-
292  C000            ;KONSTANTEN IM ROM. JEWEILS ANGABE
293  C000            ;VON LOW- UND HIGH-BYTE
294  C000            ;
300  C000            F1L=$BC
310  C000            F1H=$B9    ; FLIESSK.KONST. 1
320  C000            F05L=$11
330  C000            F05H=$BF   ; FLIESSK.KONST. 0.5
340  C000            F2PIL=$09
350  C000            F2PIH=$E3 ; FLIESSK.KONST. 2 * PI
360  C000            ;
```

```
500  C000          ;*******************************
510  C000          ;* ROM - ROUTINEN              *
520  C000          ;*******************************
525  C000          ;
530  C000          GETBYT=$B79B ; LIES BYTE-WERT
531  C000          ;GETBYT LIEST BYTE-PARAMETER AUS
532  C000          ;BASIC-TEXT. WERT STEHT ANSCHLIESSEND
533  C000          ;IM X-REGISTER.
534  C000          ;
540  C000          GETADR=$B7F7 ; LIES INTEGER-WERT
541  C000          ;GETADR LIEST ADRESS-PARAMETER AUS
542  C000          ;BASIC-TEXT. WERT STEHT ANSCHLIESSEND
543  C000          ;IM ADRL,ADRH.
544  C000          ;
550  C000          GETAB=$B7EB  ; GETADR & GETBYT
551  C000          ;KOMBINIERTE ROUTINE:
552  C000          ;GETADR & CHKCOM & GETBYT
553  C000          ;
560  C000          FRMNUM=$AD8A ; LIES FLOAT.P.-WERT
561  C000          ;FRMNUM LIEST FLIESSKOMMA-PARAMETER A
562  C000          ;BASIC-TEXT. WERT STEHT ANSCHLIESSEND
563  C000          ;IM FLIESSKOMMA-AKKU (FAC).
564  C000          ;
570  C000          CHKCOM=$AEFD ; KOMMA UEBERLESEN
571  C000          ;CHKCOM UEBERLIEST EIN KOMMA IM BASIC
572  C000          ;WENN KEIN KOMMA, DANN SYNTAX ERROR.
573  C000          ;
580  C000          ILQERR=$B248 ; ILL. QUANTITY ERROR
581  C000          ;AUSGABE VON 'ILLEGAL QUANTITY ERROR'
582  C000          ;
590  C000          DIZERR=$BB8A ; ZERODIVIDE ERROR
591  C000          ;AUSGABE VON 'DIVISION BY ZERO ERROR'
592  C000          ;
600  C000          FLDFACK=$BBA2; KONSTANTE NACH FAC
601  C000          ;FLDFACK LAEDT FAC MIT KONSTANTE.
602  C000          ;EINGABEPARAMETER:
603  C000          ; AKKU = LOW-BYTE KONSTANTENADRESSE
604  C000          ; X-REG= HIGH-BYTE KONSTANTENADRESSE
605  C000          ;
610  C000          FPLUSK=$B867 ; FAC = FAC+KONSTANTE
611  C000          ;PARAMETER SIEHE FLDFACK
612  C000          ;
620  C000          FSIN=$E26B   ; FAC = SIN(FAC)
621  C000          ;BASIC-FUNKTION SIN
622  C000          ;
630  C000          FCOS=$E264   ; FAC = COS(FAC)
631  C000          ;BASIC-FUNKTION SIN
632  C000          ;
640  C000          FACXY=$BBD4  ; FAC NACH ( X / Y )
641  C000          ;FACXY BRINGT FAC NACH VARIABLE
642  C000          ;EINGABEPARAMTER:
```

```
643  C000            ; X-REG = LOW -BYTE VARAIABLENADRESSE
644  C000            ; Y-REG = HIGH-BYTE VARAIABLENADRESSE
645  C000            ;
650  C000            FCHS=$BFD4    ; FAC = -FAC
651  C000            ; WECHSELT NUR DAS VORZEICHEN VON FAC
652  C000            ;
660  C000            FKMINUS=$B850; FAC = KONSTANTE-FAC
661  C000            ;PARAMETER SIEHE FLDFACK
662  C000            ;
670  C000            FMALK=$BA28   ; FAC = FAC*KONSTANTE
671  C000            ;PARAMETER SIEHE FLDFACK
672  C000            ;
680  C000            FKDIV=$BB0F   ; FAC = KONSTANTE/FAC
681  C000            ;PARAMETER SIEHE FLDFACK
682  C000            ;
690  C000            FSGN=$BC2B    ;  A = SGN(FAC)
691  C000            ;BRINGT VORZEICHEN VON FAC IN AKKU
692  C000            ;UND SETZT CARRY- UND ZERO-FLAG
693  C000            ;WENN FAC FREITIV,
694  C000            ;DANN AKKU=1 , CARRY=1 , ZERO=0
695  C000            ;WENN FAC = 0,
696  C000            ;DANN AKKU=0 , CARRY=1 , ZERO=1
697  C000            ;WENN FAC NEGATIV,
698  C000            ;DANN AKKU=$FF, CARRY=0 , ZERO=0
699  C000            ;
700  C000            FLPINT=$BC9B  ;FAC NACH INT
701  C000            ;BRINGT GANZZAHLIGEN TEIL DES
702  C000            ;FAC NACH INTL,INTH .
703  C000            ;
```

Die Startadresse der hier beschriebenen Grafik-Routinen wurde auf $C000 = 49152 gelegt. Dort ist im Commodore 64 ein von Basic nicht benützter RAM-Bereich. Die Bedeutung der verwendeten Konstanten und ROM-Routinen entnehmen Sie bitte dem obenstehenden Listing, das an dieser Stelle für sich selbst spricht.

6.2.2 Sprungtabelle und Hilfszellen

```
1000  C000            ;*******************************
1010  C000            ;*  SPRUNGTABELLE              *
1020  C000            ;*******************************
1030  C000  4C        BYT $4C ;JUMP
1040  C001  3080    R WOR PUNKT
1050  C003  4C        BYT $4C ;JUMP
1060  C004  3080    R WOR GREIN
1070  C006  4C        BYT $4C ;JUMP
1080  C007  9080    R WOR LOESCH
1090  C009  4C        BYT $4C ;JUMP
1100  C00A  8080    R WOR FARBE
1110  C00C  4C        BYT $4C ;JUMP
```

```
1120  C00D  8080    R WOR GRAUS
1130  C00F  4C        BYT $4C ;JUMP
1140  C010  8080    R WOR LINIE
1150  C012  4C        BYT $4C ;JUMP
1160  C013  8080    R WOR RECHT
1170  C015  4C        BYT $4C ;JUMP
1180  C016  8080    R WOR BLOCK
1190  C018  4C        BYT $4C ;JUMP
1200  C019  8080    R WOR ELLIPSE
1210  C01B  4C        BYT $4C ;JUMP
1220  C01C  8080    R WOR RADIUS
1300  C01E            ;*******************************
1310  C01E            ;* HILFSZELLEN (VARIABLE)      *
1320  C01E            ;*******************************
1330  C01E            BASH  = * / 256
1340  C01E            BASIS = BASH * 256
1350  C01E            ;
1360  C01E            ;FLIESSKOMMAVARIABLE (JE 5 BYTE):
1370  C01E            ;
1380  C01E  00        RX: BYT 0 ;RADIUS X-RICHTUNG
1390  C01F  0000      WOR 0
1400  C021  0000      WOR 0
1410  C023  00        RY: BYT 0 ;RADIUS Y-RICHTUNG
1420  C024  0000      WOR 0
1430  C026  0000      WOR 0
1440  C028  00        LAUF: BYT 0 ;LAUFVARIABLE
1450  C029  0000      WOR 0
1460  C02B  0000      WOR 0
1470  C02D  00        STEP: BYT 0 ;SCHRITTWEITE
1480  C02E  0000      WOR 0
1490  C030  0000      WOR 0
1500  C032  00        WINK: BYT 0 ;WINKEL
1510  C033  0000      WOR 0
1520  C035  0000      WOR 0
1530  C037            ;
1540  C037            ;BYTE- UND WORD-VARAIBLE:
1550  C037            ;
1560  C037  00        SPEICH1: BYT 0;HILFSZELLE ZUM
1561  C038            ;SPEICHERN VON BYTE 17 DES VIC
1570  C038  00        SPEICH2: BYT 0;HILFSZELLE ZUM
1571  C039            ;SPEICHERN VON BYTE 24 DES VIC
1580  C039  0000      AX: WOR 0      ; KOORDINATEN
1590  C03B  00        AY: BYT 0      ; ANFANGSPUNKT
1600  C03C  0000      EX: WOR 0      ; KOORDINATEN
1610  C03E  00        EY: BYT 0      ; ENDPUNKT
1620  C03F  0000      XDIFF: WOR 0   ; DIFFERENZ X-KOORD.
1630  C041  00        YDISGN: BYT 0 ; SGN. DIFF. Y-KOORD.
1631  C042            ;YDISGN = $FF , WENN AY > EY
1632  C042            ;YDISGN = 0   , WENN AY = EY
1633  C042            ;YDISGN = 1   , WENN AY < EY
1640  C042  00        YDIABS: BYT 0 ; ABS. DIFF. Y-KOORD.
```

```
1641  C043            ;YDIABS = ABS(EY-AY)
1650  C043  0000      XLAUF: WOR 0   ; LAUFVARIABLE
1660  C045  00        YLAUF: BYT 0   ; LAUFVARIABLE
1670  C046  0000      QX: WOR 0      ; KOORDINATEN
1680  C048  00        QY: BYT 0      ;  OBERER PUNKT
1690  C049  0000      UX: WOR 0      ; KOORDINATEN
1700  C04B  00        UY: BYT 0      ;  UNTERER PUNKT
1710  C04C  0000      MX: WOR 0      ; KOORDINATEN
1720  C04E  00        MY: BYT 0      ;  KREISMITTELPUNKT
```

Am Anfang des Programms steht eine Sprungtabelle, mit Sprüngen auf alle wichtigen Programmstücke. Dieses Verfahren mag umständlich erscheinen, daß man zunächst auf eine Programmzelle springt, in der wieder ein Sprung auf eine andere Zelle steht, jedoch hat sich gezeigt, daß es bei mittleren und größeren Programmkomplexen notwendig ist, eine konstante Adresse zu haben, über die ein Unterprogramm angesprochen werden kann. Wenn man dann ein Unterprogramm ändert, können sich dadurch die tatsächlichen Programmbeginne der folgenden Unterprogramme, die im gleichen Komplex stehen, ändern. Man müßte sonst auch alle externen Aufrufe dieser Unterprogramme ändern. Hat man jedoch eine Sprungtabelle, so braucht nur dieser Programmkomplex neu übersetzt zu werden und die Sprungtabelle korrigiert sich selbst.

Die Sprungtabelle wird jeweils so realisiert, daß als erstes ein Byte $4C, das dem JMP-Befehl entspricht, gefolgt von einem Wort, das das Sprungziel enthält, angelegt werden.

Im Adressbereich $C01E bis $C04E werden Hilfszellen und Variablen des Assemblerprogramms definiert. Die Bedeutung der Variablen entnehmen Sie bitte dem Listing. Es sei hier nur angemerkt, daß hier drei verschiedene Typen von Variablen oder Hilfszellen verwendet werden. Dies sind zum einen die Fließkomma-Variablen die je 5 Byte benötigen, zum anderen Wort-Variablen die aus zwei Byte bestehen, und schließlich Byte-Variablen die nur aus einem einzelnen Byte bestehen.

6.2.3 Multiplizieren und Dividieren

In den Befehlen PUNKT und LINIE, werden jeweils arithmetische Berechnungen benötigt. Im Gegensatz zu Subtrahieren und Addieren, die durch einfache Maschinenbefehle zu realisieren sind, braucht man für die Bewältigung einer Mul-

tiplikation oder einer Division ein kleines Unterprogramm. Die im Basic-Interpreter vorhandenen Routinen sind hier ungeeignet, weil sie nur Fließkommazahlen verarbeiten und die zweimalige Umwandlung zusammen mit der Fließkomma-Arithmetik zu viel Zeit beanspruchen würde.

Multiplizieren

```
10070  C3F8            MULT:
10080  C3F8  A900      LDA #$00
10090  C3FA  8557      STA PRDL
10100  C3FC  8558      STA PRDH : PRD = 0
10110  C3FE  A008      LDY #$08 ; SCHLEIFENZAEHLER
10120  C400  18        CLC
10130  C401            MULT1:
10140  C401  265B      ROL FAK2
10150  C403  9000   R  BCC MULT2 ; NICHTS ADDIEREN
10160  C405  18        CLC
10170  C406  A557      LDA PRDL
10180  C408  6559      ADC FAK1L
10190  C40A  8557      STA PRDL
10200  C40C  A558      LDA PRDH
10210  C40E  655A      ADC FAK1H
10220  C410  8558      STA PRDH ; PRD = PRD + FAK1
10230  C412            MULT2:
10240  C412  88        DEY
10250  C413  F000   R  BEQ MULEND ; 8 BIT VERARBEITET
10260  C415  2657      ROL PRDL
10270  C417  2658      ROL PRDH : PRD VERDOPPELN
10280  C419  90E6      BCC MULT1 ;CARRY MUSS 0 SEIN
10290  C41B            MULEND:
10300  C41B  60        RTS ; ENDE UPRO MULTIPLIZIEREN
```

Dieses Unterprogramm multipliziert eine 16 Bit-Zahl die in den Speicherzellen FAK1L, FAK1H abgelegt ist, mit einer 8 Bit-Zahl in FAK2. Das Produkt der beiden Zahlen wird in den Speicherzellen PRDL, PRDH abgelegt. Bei dem Algorithmus handelt es sich hier um die übliche bitweise Multiplikation, die abgeleitet werden kann, aus dem Verfahren, das man anwendet, wenn man mit Bleistift und Papier multipliziert.

Dividieren

```
11070  C41C            DIV:
11080  C41C  A900      LDA #0
```

```
11090  C41E  855C      STA QUOTL
11100  C420  855D      STA QUOTH ; FAK2 = 0
11110  C422  A559      LDA FAK1L
11120  C424  D000   R  BNE DIV1
11130  C426  A55A      LDA FAK1H
11140  C428  D000   R  BNE DIV1
11150  C42A  4C8ABB    JMP DIZERR ; DIVISOR = 0
11160  C42D             ;
11170  C42D             DIV1:
11180  C42D  A558      LDA PRDH
11190  C42F  D000   R  BNE DIV2   ; PRD > 0
11200  C431  A557      LDA PRDL
11210  C433  F000   R  BEQ DIVEND ; PRD = 0
11220  C435             DIV2:
11230  C435  38        SEC
11240  C436  A557      LDA PRDL
11250  C438  E559      SBC FAK1L
11260  C43A  8557      STA PRDL
11270  C43C  A558      LDA PRDH
11280  C43E  E55A      SBC FAK1H
11290  C440  8558      STA PRDH ; PRD = PRD - FAK1
11300  C442  9000   R  BCC DIVEND ; PRD < 0
11310  C444  E65C      INC QUOTL ; FAK2 = FAK2 + 1
11320  C446  D0E5      BNE DIV1 ; KEIN UEBERTRAG
11330  C448  E65D      INC QUOTH;UEBERTRAG SPEICHERN
11340  C44A  D0E1      BNE DIV1 ; SCHLEIFE BIS PRD <= 0
11350  C44C             DIVEND:
11360  C44C  60        RTS ; ENDE DIVIDIEREN
```

Dieses Programm teilt eine 16 Bit-Zahl, die in den Speicherzellen PRDL, PRDH übergeben wird, durch eine 16 Bit-Zahl in den Zellen FAK1L, FAK1H, und liefert als Ergebnis eine 16 Bit-Zahl, die in die Speicherzellen QUOTL, QUOTH abgelegt wird.

Wenn der Divisor gleich Null ist, wird zur Ausgabe der Fehlermeldung "DIVISION BY ZERO ERROR" gesprungen. Im Übrigen besteht der Algorithmus aus einem fortgesetzten Abziehen des Divisors vom Dividenden und mitzählen der Schleifendurchläufe, bis der Dividend kleiner oder gleich Null ist. Der Algorithmus, wie er von Rechnern ausgeführt wird, ist zwar etwas eleganter und schneller, doch für unseren Umfang etwas zu komplex.

6.2.4 GREIN - Grafik einschalten

```
4050  C080             GREIN:
4060  C080  209BB7     JSR GETBYT: ZEICHENFARBE -> X
```

```
4070  C0B3  8A        TXA
4080  C0B4  0A        ASL A
4090  C0B5  0A        ASL A
4100  C0B6  0A        ASL A
4110  C0B7  0A        ASL A ; A = ZF * 16
4120  C0B8  8D37C0    STA SPEICH1
4130  C0BB  209BB7    JSR GETBYT; HI.GR.FARBE -> X
4140  C0BE  8A        TXA
4150  C0BF  0D37C0    ORA SPEICH1 ;A = ZF * 16 + HF
4160  C0C2  AA        TAX
4170  C0C3  208080 R  JSR FARBE ;VIDEO-RAM BESCHREIBEN
4180  C0C6  AD11D0    LDA VIC+17
4190  C0C9  8D37C0    STA SPEICH1 ; ALTE BITS SICHERN
4200  C0CC  AD18D0    LDA VIC+24
4210  C0CF  8D38C0    STA SPEICH2 ; ALTE BITS SICHERN
4220  C0D2  A93B      LDA #%00111011
4230  C0D4  8D11D0    STA VIC+17  ; HI-RES EIN
4240  C0D7  A918      LDA #%00011000
4250  C0D9  8D18D0    STA VIC+24  ; GRAPHIK-RAM BEI $4000
4260  C0DC  208080 R  JSR LOESCH  ; GRAPHIK LOESCHEN
4270  C0DF  60        RTS ; ENDE UPRO GRAPHIK EIN
4500  C0E0            ;*******************************
4510  C0E0            ;*                             *
4520  C0E0            ;* GRAPHIK LOESCHEN            *
4530  C0E0            ;*                             *
4540  C0E0            ;*******************************
4550  C0E0            LOESCH:
4560  C0E0  A920      LDA #GANFH
4570  C0E2  85FE      STA Z+1
4580  C0E4  A900      LDA #0
4590  C0E6  85FD      STA Z ; Z = GANF
4600  C0E8            LOE1:
4610  C0E8  A8        TAY ; Y = 0
4620  C0E9            LOE2:
4630  C0E9  91FD      STA (Z),Y
4640  C0EB  C8        INY
4650  C0EC  D0FB      BNE LOE2 ;SCHLEIFE LAUFVAR. Y
4660  C0EE  E6FE      INC Z+1
4670  C0F0  A4FE      LDY Z+1
4680  C0F2  C040      CPY #GENDH ; ENDEKRITERIUM
4690  C0F4  D0F2      BNE LOE1 ;SCHLEIFE LAUFVAR (Z+1)
4700  C0F6  60        RTS ;ENDE UPRO GRAPHIK LOESCHEN
5000  C0F7            ;*******************************
5010  C0F7            ;*                             *
5020  C0F7            ;* FARBEN SETZEN               *
5030  C0F7            ;*                             *
5040  C0F7            ;*******************************
5050  C0F7            FARBE:
5060  C0F7  8A        TXA ; X ENTHIELT FARBEN
5070  C0F8  A000      LDY #0; SCHLEIFENZAEHLER=0
5080  C0FA            FARB1:
```

```
5090  C0FA  990004    STA VANFANG,Y      ;BYTE 0-249
5100  C0FD  99FA04    STA VANFANG+250,Y  ;BYTE 250-499
5110  C100  99F405    STA VANFANG+500,Y  ;BYTE 500-749
5120  C103  99EE06    STA VANFANG+750,Y  ;BYTE 750-999
5130  C106  C8        INY
5140  C107  C0FA      CPY #250
5150  C109  D0EF      BNE FARB1 ;SCHLEIFE BIS Y=250
5160  C10B  60        RTS ;ENDE UPRO FARBEN SETZEN
```

Das Programm zum Einschalten der Grafik wird zusammen mit den Unterprogrammen zum Löschen und Farbesetzen behandelt, die auch vom Programmstück GREIN aufgerufen werden.

Im Unterprogramm Grafik einschalten (GREIN) wird zuerst mit Hilfe der Routine GETBYT ein Byte-Wert aus dem Basic-Text eingelesen, mit 16 multipliziert und zwischengespeichert. Dann wird nochmal die Routine GETBYT aufgerufen und der erhaltene Wert zu dem vorher zwischengespeicherten addiert und in das X-Register gebracht. Das X-Register ist Eingabeparameter für das Unterprogramm Farbe, das jetzt aufgerufen wird. Dann werden noch die aktuellen Werte zweier Bytes im Video-Controller gesichert und mit den notwendigen Werten für hochauflösende Grafik beschrieben. Schließlich wird noch der Grafikspeicher gelöscht (Unterprogrammaufruf LÖSCH) und das Unterprogramm verlassen.

Grafik löschen

In diesem Unterprogramm werden alle Speicherzellen des Grafikspeichers gelöscht, d.h. auf '0' gesetzt. Als Laufvariable dient dabei das Zellenpaar Z,Z+1.

Farben setzen

In diesem Unterprogramm wird einfach der gesamte Videospeicher mit dem Wert des X-Registers beschrieben.

Aufruf von GREIN:

Adresse bei Aufruf mit SYS: 49155

Syntax: GREIN,ZF,HF bzw. SYS49155,ZF,HF

Ein kurzes Beispiel:

```
1010 GREIN,7,6          für gelbe Punkte auf blauem Grund
```

6.2.5 GRAUS - Grafik ausschalten

```
5550  C10C            GRAUS:
5560  C10C  AD37C0    LDA SPEICH1
5570  C10F  8D11D0    STA VIC+17
5580  C112  AD38C0    LDA SPEICH2 ;ALTE BITS IM VIC
5590  C115  8D18D0    STA VIC+24  ;WIEDERHERSTELLEN
5600  C118  60        RTS ; ENDE UPRO GRAPHIK AUS
```

Dieses kurze Unterprogramm schreibt lediglich die vorher beim Einschalten der Grafik zwischengespeicherten Werte wieder in den Video-Controller zurück, so daß die Grafik wieder ausgeschaltet wird.

Adresse bei Aufruf mit SYS: 49164

Syntax: GRAUS bzw. SYS49164

6.2.6 PUNKT

```
3060  C04F            PUNKT:
3070  C04F            ;
3080  C04F  20FDAE    JSR CHKCOM
3090  C052  20EBB7    JSR GETAB ;KOORDINATEN LESEN
3100  C055            ;
3110  C055            PUNKS:
3120  C055            ; X-KOORD. IN ADRL/ADRH
3130  C055            ; Y-KOORD. IM X-REGISTER
3140  C055            ;
3150  C055  E0C8      CPX #200
3160  C057  B000   R  BCS PUNERR ;Y-KOORD. > 199
3170  C059  A515      LDA ADRH
3180  C05B  F000   R  BEQ PUN1   ;X-KOORD. < 256
3190  C05D  C901      CMP #1
3200  C05F  D000   R  BNE PUNERR ;X-KOORD. > 511
3210  C061  A514      LDA ADRL
3220  C063  C940      CMP #64
3230  C065  B000   R  BCS PUNERR ;X-KOORD. > 319
3240  C067            PUN1:
3250  C067  A514      LDA ADRL
3260  C069  2907      AND #$07
3270  C06B  A8        TAY ; Y = X AND 7
3280  C06C  38        SEC ; ZUM HINEINROTIEREN
3290  C06D  A900      LDA #$00
3300  C06F            PUN2:
3310  C06F  6A        ROR A
```

```
3320  C070  88        DEY
3330  C071  10FC      BPL PUN2
3340  C073            ;A = 2 ↑ ( 7 - (XK AND 7) )
3350  C073  48        PHA ; WERT SICHERN
3360  C074            ;
3370  C074  A514      LDA ADRL
3380  C076  29F8      AND #$F8
3390  C078  8514      STA ADRL ; ADR = XK AND $FFF8
3400  C07A  8A        TXA
3410  C07B  2907      AND #$07 ; A = YK AND 7
3420  C07D  18        CLC
3430  C07E  6514      ADC ADRL ; A = ADR + (YK AND 7)
3440  C080  8514      STA ADRL
3450  C082  A515      LDA ADRH
3460  C084  6920      ADC #GANFH
3470  C086  8515      STA ADRH
3480  C088            ;ADR= GANF+(XK AND $FFF8)+(YK AND 7)
3490  C088  8A        TXA
3500  C089  29F8      AND #$F8
3510  C08B  8559      STA FAK1L
3520  C08D  A900      LDA #$00
3530  C08F  855A      STA FAK1H ;FAK1 = YK AND $00F8
3540  C091  A928      LDA #40
3550  C093  855B      STA FAK2  ;FAK2 = 40
3560  C095  208080 R  JSR MULT  ;PRD = FAK1 * FAK2
3570  C098  18        CLC
3580  C099  A514      LDA ADRL
3590  C09B  6557      ADC PRDL
3600  C09D  8514      STA ADRL
3610  C09F  A515      LDA ADRH
3620  C0A1  6558      ADC PRDH
3630  C0A3  8515      STA ADRH  ;ADR = ADR + PRD
3640  C0A5            ;ADR= GANF+(XK AND $FFF8) +
3650  C0A5            ;  + (YK AND 7) + 40 * (YK AND $F8)
3660  C0A5  68        PLA ; WERT HOLEN (ZU SETZENDES BIT)
3670  C0A6  A000      LDY #$00
3680  C0A8  1114      ORA (ADRL),Y ; MIT ZELLE ODERIEREN
3690  C0AA  9114      STA (ADRL),Y ; UND SPEICHERN
3700  C0AC            PUNEND:
3710  C0AC  60        RTS  ; ENDE UPRO PUNKT SETZEN
3720  C0AD            PUNERR:
3730  C0AD  4C48B2    JMP ILQERR ;WENN X>319 ODER Y>199
```

Dieses Unterprogramm hat zwei Einsprungstellen. Die erste (Label PUNKT) wird benützt, wenn das Unterprogramm als Basic-Befehl benützt wird. Die andere Einsprungstelle (PUNKS) wird innerhalb der folgenden Unterprogramme zum Setzen eines einzelnen Punktes verwendet.

Bei Verwendung des Labels PUNKT muß beachtet werden, daß

die Parameter (X- und Y-Koordinate des Punktes) aus dem Basic-Text mit Hilfe der Routine GETAB eingelesen werden und deshalb anschließend die X-Koordinate in den Zellen ADRL-ADRH abgespeichert ist und die Y-Koordinate sich im X-Register befindet. Genau dort müssen also auch die Koordinaten abgelegt werden, wenn das Unterprogramm mit Hilfe des Labels PUNKS aufgerufen wird.

Im Bereich zwischen den Labels PUNKS und PUN1 durchlaufen die Koordinaten eine kurze Plausibilitätsprüfung. Wenn die Koordinaten zu groß sind, springt das Programm zur Ausgabe von 'ILLEGAL QUANTITY ERROR'.

Die Berechnung der effektiven Adresse und des zu setzenden Bits geschieht analog zu dem in Kapitel 4.1.2 beschriebenen Basic-Programm. Genauere Angaben zum Algorithmus entnehmen Sie bitte dem Listing.

Adresse bei Aufruf mit SYS: 49152

Syntax: PUNKT,PX,PY bzw. SYS49152,PX,PY

Ein kurzes Beispiel:

```
1100 REM ***PUNKTE SETZEN ***
1110 PUNKT,1,1
1120 PUNKT,10,10
1130 PUNKT,165,100
1140 PUNKT,166,100
1150 PUNKT,167,160
```

6.2.7 LINIE

```
6035  C119             LINIE:
6040  C119  20FDAE      JSR CHKCOM
6045  C11C  20EBB7      JSR GETAB  ; AX,AY LESEN
6050  C11F  8E3BC0      STX AY     ; AY SPEICHERN
6055  C122  A514        LDA ADRL
6060  C124  8D39C0      STA AX
6065  C127  A515        LDA ADRH
6070  C129  8D3AC0      STA AX+1  ; AX GESPEICHERT
6075  C12C  20FDAE      JSR CHKCOM
6080  C12F  20EBB7      JSR GETAB  ; EX,EY LESEN
6085  C132  8E3EC0      STX EY     ; EY SPEICHERN
6090  C135  A614        LDX ADRL
6095  C137  8E3CC0      STX EX
6100  C13A  A515        LDA ADRH
6105  C13C  8D3DC0      STA EX+1   ; EX GESPEICHERT
6110  C13F              ;
```

```
6115  C13F             LINIES:
6120  C13F             ; AX,AY : ANFANGSPUNKT
6125  C13F             ; EX,EY : ENDPUNKT
6130  C13F             ;
6135  C13F  AE3CC0     LDX EX
6140  C142  AD3DC0     LDA EX+1
6145  C145  CD3AC0     CMP AX+1
6150  C148  F000    R  BEQ LIN1
6155  C14A  B000    R  BCS LIN2 ; AX < EX
6160  C14C             LIN1:
6165  C14C  EC39C0     CPX AX
6170  C14F  F000    R  BEQ LINXGL ; AX = EX
6175  C151  B000    R  BCS LIN2 ; AX < EX
6180  C153  AD39C0     LDA AX     ;VERTAUSCHEN
6185  C156  AE3CC0     LDX EX     ;VON ANFANGS-
6190  C159  8D3CC0     STA EX     ;UND ENDPUNKT
6195  C15C  8E39C0     STX AX
6200  C15F  AD3AC0     LDA AX+1
6205  C162  AE3DC0     LDX EX+1
6210  C165  8D3DC0     STA EX+1
6215  C168  8E3AC0     STX AX+1
6220  C16B  AD3BC0     LDA AY
6225  C16E  AE3EC0     LDX EY
6230  C171  8D3EC0     STA EY
6235  C174  8E3BC0     STX AY
6240  C177             LIN2:     ; HIER IST AX < EX
6245  C177  38         SEC
6250  C178  AD3CC0     LDA EX
6255  C17B  ED39C0     SBC AX
6260  C17E  8D3FC0     STA XDIFF
6265  C181  AD3DC0     LDA EX+1
6270  C184  ED3AC0     SBC AX+1
6275  C187  8D40C0     STA XDIFF+1 ; XDIFF = EX-AX
6280  C18A  A200       LDX #0
6285  C18C  8E41C0     STX YDISGN; YDISGN = 0
6290  C18F  38         SEC
6295  C190  AD3EC0     LDA EY
6300  C193  ED3BC0     SBC AY
6305  C196  F000    R  BEQ LINYGL; EY = AY
6310  C198  EE41C0     INC YDISGN ;YDISGN=SGN(EY-AY) =1
6315  C19B  8D42C0     STA YDIABS ;ABS(EY-AY)
6320  C19E  B000    R  BCS LIN3    ; EY > AY
6325  C1A0  49FF       EOR #$FF    ; AKKU KOMPLEMENTIEREN
6330  C1A2  6901       ADC #1      ; (2-ER KOMPLEMENT)
6335  C1A4  CA         DEX         ; X = $FF
6340  C1A5  8E41C0     STX YDISGN ;YDISGN=SGN(EY-AY) =-1
6345  C1A8  8D42C0     STA YDIABS ; YDIABS = ABS(YDIFF)
6350  C1AB             LIN3:
6355  C1AB  AD40C0     LDA XDIFF+1
6360  C1AE  D000    R  BNE LINX  ; XDIFF > YDIABS
6365  C1B0  AD3FC0     LDA XDIFF
```

```
6370  C1B3  CD42C0    CMP YDIABS
6375  C1B6  B000    R BCS LINX   : XDIFF >= YDIABS
6380  C1B8  4C8080  R JMP LINY   ; YDIABS > XDIFF
6385  C1BB            ;
6390  C1BB            LINXGL:
6395  C1BB  AD3EC0    LDA EY
6400  C1BE  CD3BC0    CMP AY
6405  C1C1  B000    R BCS LINXGL1 ; AY <= EY
6410  C1C3  AE3BC0    LDX AY
6415  C1C6  8E3EC0    STX EY
6420  C1C9  8D3BC0    STA AY ; AY MIT EY VERTAUSCHT
6425  C1CC            LINXGL1:
6430  C1CC  AE3BC0    LDX AY
6435  C1CF            LINXGL2:
6440  C1CF  8A        TXA  ; X = Y-KOORD.
6445  C1D0  48        PHA  ; SICHERN
6450  C1D1  AD39C0    LDA AX
6455  C1D4  8514      STA ADRL
6460  C1D6  AD3AC0    LDA AX+1
6465  C1D9  8515      STA ADRH ; ADR = X-KOORD.
6470  C1DB  2055C0    JSR PUNKS ; PUNKT SETZEN
6475  C1DE  68        PLA  ; Y-KOORD. HOLEN
6480  C1DF  AA        TAX
6485  C1E0  E8        INX  ; NEXT Y-KOORD.
6490  C1E1  EC3EC0    CPX EY ;ENDPUNKT ERREICHT PRINT
6495  C1E4  90E9      BCC LINXGL2 ; NEIN
6500  C1E6  60        RTS ; ENDE
6505  C1E7            ;
6510  C1E7            LINYGL:
6515  C1E7  AD39C0    LDA AX
6520  C1EA  8514      STA ADRL
6525  C1EC  AD3AC0    LDA AX+1
6530  C1EF  8515      STA ADRH ; ADR = X-KOORD. (AX)
6535  C1F1  AE3EC0    LDX EY   ;  X  = Y-KOORD.
6540  C1F4  2055C0    JSR PUNKS ; PUNKT SETZEN
6545  C1F7  EE39C0    INC AX
6550  C1FA  D000    R BNE LINYGL1 ; KEIN UEBERTRAG
6555  C1FC  EE3AC0    INC AX+1
6560  C1FF            ; AX UM 1 ERHOEHT
6565  C1FF            LINYGL1:
6570  C1FF  AD3DC0    LDA EX+1
6575  C202  CD3AC0    CMP AX+1
6580  C205  9000    P BCC LINEND ;ENDPUNKT ERREICHT
6585  C207  D0DE      BNE LINYGL ;NOCH NICHT ERREICHT
6590  C209  AD3CC0    LDA EX       ; VERGLEICH LOW-BYTE
6595  C20C  CD39C0    CMP AX
6600  C20F  B0D6      BCS LINYGL  ;NOCH NICHT ERREICHT
6605  C211            LINEND:
6610  C211  60        RTS  ; ENDE
6615  C212            ;
6620  C212            LINI:
```

```
6625  C212  A900      LDA #0
6630  C214  8D43C0    STA XLAUF
6635  C217  8D44C0    STA XLAUF+1 ; XLAUF = 0
6640  C21A            LINX1:
6645  C21A  18        CLC
6650  C21B  AD43C0    LDA XLAUF
6655  C21E  8559      STA FAK1L
6660  C220  6D39C0    ADC AX
6665  C223  8514      STA ADRL
6670  C225  AD44C0    LDA XLAUF+1
6675  C228  855A      STA FAK1H   ; FAK1 = XLAUF
6680  C22A  6D3AC0    ADC AX+1
6685  C22D  8515      STA ADRH    ; ADR = AX + XLAUF
6690  C22F  AD42C0    LDA YDIABS
6695  C232  855B      STA FAK2    ; FAK2 = YDIFF = ABS(EY-AY)
6700  C234  208080 R  JSR MULT    ; PRD = XLAUF * YDIFF
6705  C237  AD3FC0    LDA XDIFF
6710  C23A  8559      STA FAK1L
6715  C23C  AD40C0    LDA XDIFF+1
6720  C23F  855A      STA FAK1H   ; FAK1 = XDIFF
6725  C241  208080 R  JSR DIV ; QUOT = XLAUF * YDIFF / XDIFF
6730  C244  AD3BC0    LDA AY
6735  C247  2C41C0    BIT YDISGN
6740  C24A  3000    R BMI LINX2 ; YDIFF NEGATIV
6745  C24C  18        CLC
6750  C24D  655C      ADC QUOTL ;ADDIEREN, WENN YDIFF>0
6755  C24F  9000    R BCC LINX3 ;UNBED. SPRUNG
6760  C251            LINX2:
6765  C251  38        SEC
6770  C252  E55C      SBC QUOTL ;SUBTRAHIEREN
6775  C254            LINX3:
6780  C254  AA        TAX ; X = AY + (XLAUF * YDIFF / XDIFF)
6785  C255  2055C0    JSR PUNKS ; PUNKT SETZEN
6790  C258  EE43C0    INC XLAUF
6795  C25B  D000    R BNE LINX4 ;KEIN UEBERTRAG
6800  C25D  EE44C0    INC XLAUF+1
6805  C260            ; XLAUF UM 1 ERHOEHT
6810  C260            LINX4:
6815  C260  AD40C0    LDA XDIFF+1
6820  C263  CD44C0    CMP XLAUF+1
6825  C266  9000    R BCC LINX5 ; ENDPUNKT ERREICHT
6830  C268  D0B0      BNE LINX1 ; NICHT ERREICHT
6835  C26A  AD3FC0    LDA XDIFF
6840  C26D  CD43C0    CMP XLAUF
6845  C270  B0A8      BCS LINX1 ; NICHT ERREICHT
6850  C272            LINX5:
6855  C272  60        RTS ; ENDE
6860  C273            ;
6865  C273            LINY:
6870  C273  A900      LDA #0
6875  C275  8D45C0    STA YLAUF ; YLAUF = 0
```

```
6880  C270            LINY1:
6885  C270  855B      STA FAK2 ; FAK2 = YLAUF
6890  C27A  AD3BC0    LDA AY
6895  C27D  2C41C0    BIT YDISGN
6900  C280  3000   R  BMI LINY2
6905  C282  18        CLC
6910  C283  6D45C0    ADC YLAUF ; YLAUF ADDIEREN
6915  C286            ;           WENN EY>AY
6920  C286  9000   R  BCC LINY3 ; UNBED. SPRUNG
6925  C288            LINY2:
6930  C288  38        SEC
6935  C289  ED45C0    SBC YLAUF ; YLAUF SUBTRAHIEREN
6940  C28C            ;           WENN AY>EY
6945  C28C            LINY3:
6950  C28C  48        PHA ; Y-KOORD. SICHERN
6955  C28D  AD3FC0    LDA XDIFF
6960  C290  8559      STA FAK1L
6965  C292  AD40C0    LDA XDIFF+1
6970  C295  855A      STA FAK1H ; FAK1=XDIFF
6975  C297  208080 R  JSR MULT ; PRD = YLAUF * XDIFF
6980  C29A  AD42C0    LDA YDIABS
6985  C29D  8559      STA FAK1L
6990  C29F  A900      LDA #0
6995  C2A1  855A      STA FAK1H ; FAK1 = ABS(EY-AY)
7000  C2A3  208080 R  JSR DIV ; QUOT=YLAUF*XDIFF/YDIABS
7005  C2A6  18        CLC
7010  C2A7  A55C      LDA QUOTL
7015  C2A9  6D39C0    ADC AX
7020  C2AC  8514      STA ADRL
7025  C2AE  A55D      LDA QUOTH
7030  C2B0  6D3AC0    ADC AX+1
7035  C2B3  8515      STA ADRH ;ADR=YLAUF*XDIFF/YDIABS + AX
7040  C2B5  68        PLA ; Y-KOORD. WIEDER HOLEN
7045  C2B6  AA        TAX ; INS X-REGISTER
7050  C2B7  2055C0    JSR PUNKS ; PUNKT SETZEN
7055  C2BA  EE45C0    INC YLAUF ; YLAUF = YLAUF +1
7060  C2BD  AD45C0    LDA YLAUF
7065  C2C0  CD42C0    CMP YDIABS ; ENDE ERREICHT PRINT
7070  C2C3  F0B3      BEQ LINY1 ; NOCH EINMAL
7075  C2C5  90B1      BCC LINY1 ; NOCH MEHRMALS
7080  C2C7  60        RTS ; ENDE
```

Das vorliegende Unterprogramm besitzt ähnlich dem Unterprogramm PUNKT, zwei Einsprungstellen.

Die erste Einsprungstelle hat das Label LINIE. Hier werden wieder die Parameter aus dem Basic-Text eingelesen und in die Speicherzellen AX, AX+1, AY, EX, EX+1 und EY abgelegt. Genau in diese Speicherzellen muß man auch die Parameter schreiben, wenn man das Unterprogramm über das Label

LINIES aufruft.

Der verwendete Algorithmus weicht etwas von dem in Kapitel 4.2.1 beschriebenen Basic-Programm ab. Deshalb soll hier kurz der Programmablauf skizziert werden. Für genauere Angaben schen Sie jedoch wieder bitte im Listing nach.

Entsprechend den Relationen zwischen Anfangs- und Endpunkt wird nach verschiedenen Programmstücken verzweigt. Als erstes wird geprüft, ob die X-Koordinaten der beiden Punkte gleich sind, dann wird zum Label LINXGL verzweigt.

Wenn die X-Koordinate des Anfangspunkts größer ist als die des Endpunkts, werden die beiden Punte vertauscht. Dies ist für das Resultat am Bildschirm unerheblich, jedoch erleichtert es die Programmierung, wenn man im folgenden davon ausgehen kann, daß die Differenz EX-AX positiv ist. Diese Differenz wird in der Speicherzelle XDIFF, XDIFF+1 zwischengespeichert. Die Differenz der Y-Koordinaten wird ebenfalls abgespeichert und zwar in der Speicherzelle YDISGN das Vorzeichen der Differenz EY-AY und in der Speicherzelle YDIABS der absolute Betrag der Differenz.

Ist diese Differenz gleich Null, wird zum Label LINYGL verzweigt.

Jetzt muß nur noch geprüft werden, ob der Absolutbetrag der Differenz EY-AY kleiner oder größer als die Differenz der X-Koordinaten ist. Wenn die Differenz der X-Koordinaten größer oder gleich dem Absolutbetrag der Y-Koordinaten ist, wird zum Label LINX verzweigt, ansonsten zum Label LINY.

LINXGL:

Da dieses Programmstück nur aufgerufen wird, wenn die X-Koordinaten der beiden Punkte gleich sind, kann man auch noch die Y-Koordinaten nach Größe sortieren, was die Verarbeitung im folgenden erleichtert. Im Wesentlichen wird nämlich in einer Schleife von Anfangswert AY bis Endwert EY alle Punkte auf einer senkrechten Linie nacheinander gezeichnet.

LINYGL:

Da die Punkte schon nach X-Koordinaten sortiert sind, kann hier dieses Programmstück sofort mit dem Anfang einer Schleife beginnen, die eine waagrechte Linie von links nach rechts zeichnet.

LINX:

Dieses Programmstück zeichnet eine Linie von links nach rechts mit einer Neigung von + 45 bis - 45 Grad. Deshalb ist es hier günstig, die X-Koordinate als Laufvariable zu benützen und daraus jeweils die aktuelle Y-Variable zu berechnen. Die mathematische Funktion hierfür lautet:

Y = AY + (XLAUF x YDIFF / XDIFF).

Zum Berechnen des Terms werden die Unterprogramme MULT und DIFF herangezogen, die in Kapitel 6.2.3 beschrieben wurden.

LINY:

Dieses Programm zeichnet schräge Linien mit Neigungswinkeln zwischen 45 und 135 Grad. Hier muß die Y-Koordinate als Laufvariable benutzt werden, und die X-Koordinate wird jeweils berechnet. Die mathematische Funktion hierfür muß also lauten:

X = AX + (YLAUF x XDIFF / YDIABS).

Adresse von LINIE bei Aufruf mit SYS: 49167

Syntax: LINIE,AX,AY,EX,EY bzw. SYS49167,AX,AY,EX,EY

Ein kurzes Beispiel:

```
1200 REM *** LINIEN ZIEHEN ***
1202 LINIE,319,199,0,0
1203 LINIE,0,199,319,0
1210 LINIE,40,40,100,40
1211 LINIE,60,25,120,25
```

6.2.8 RECHT - Rechteck zeichnen

```
8070  C2C8            RECHT:
8080  C2C8  20FDAE    JSR CHKCOM
8090  C2CB  20EBB7    JSR GETAB ; OX,OY LESEN
8100  C2CE  8E48C0    STX OY    ; OY SPEICHERN
8110  C2D1  A514      LDA ADRL
8120  C2D3  8D46C0    STA OX
8130  C2D6  A515      LDA ADRH
8140  C2D8  8D47C0    STA OX+1  ; OX GESPEICHERT
8150  C2DB  20FDAE    JSR CHKCOM
8160  C2DE  20EBB7    JSR GETAB ; UX,UY LESEN
```

```
8170  C2E1  8E4BC0   STX UY     ; UY SPEICHERN
8180  C2E4  A614     LDX ADRL
8190  C2E6  8E49C0   STX UX
8200  C2E9  A515     LDA ADRH
8210  C2EB  8D4AC0   STA UX+1   ; UX GESPEICHERT
8220  C2EE            ;
8230  C2EE            RECHS:
8240  C2EE            ; OX,OY KOORD. V. OBEREM ECKPUNKT
8250  C2EE            ; UX,UY KOORD. V. UNTEREM ECKPUNKT
8260  C2EE            ;
8270  C2EE            ;
8280  C2EE            ; LINIE ( OX , OY , UX , OY )
8290  C2EE            ;
8300  C2EE  AD46C0   LDA OX
8310  C2F1  8D39C0   STA AX
8320  C2F4  AD47C0   LDA OX+1
8330  C2F7  8D3AC0   STA AX+1
8340  C2FA  AD48C0   LDA OY
8350  C2FD  8D3BC0   STA AY
8360  C300  AD49C0   LDA UX
8370  C303  8D3CC0   STA EX
8380  C306  AD4AC0   LDA UX+1
8390  C309  8D3DC0   STA EX+1
8400  C30C  AD48C0   LDA OY
8410  C30F  8D3EC0   STA EY
8420  C312  203FC1   JSR LINIES ; LINIE ZIEHEN
8430  C315            ;
8440  C315            ; LINIE ( UX , OY , UX , UY )
8450  C315            ;
8460  C315  AD49C0   LDA UX
8470  C318  8D39C0   STA AX
8480  C31B  AD4AC0   LDA UX+1
8490  C31E  8D3AC0   STA AX+1
8500  C321  AD48C0   LDA OY
8510  C324  8D3BC0   STA AY
8520  C327  AD49C0   LDA UX
8530  C32A  8D3CC0   STA EX
8540  C32D  AD4AC0   LDA UX+1
8550  C330  8D3DC0   STA EX+1
8560  C333  AD4BC0   LDA UY
8570  C336  8D3EC0   STA EY
8580  C339  203FC1   JSR LINIES ; LINIE ZIEHEN
8590  C33C            ;
8600  C33C            ; LINIE ( UX , UY , OX , UY )
8610  C33C            ;
8620  C33C  AD49C0   LDA UX
8630  C33F  8D39C0   STA AX
8640  C342  AD4AC0   LDA UX+1
8650  C345  8D3AC0   STA AX+1
8660  C348  AD4BC0   LDA UY
8670  C34B  8D3BC0   STA AY
```

```
8580  C34E  AD46C0    LDA OX
8590  C351  8D3CC0    STA EX
8700  C354  AD47C0    LDA OX+1
8710  C357  8D3DC0    STA EX+1
8720  C35A  AD4BC0    LDA UY
8730  C35D  8D3EC0    STA EY
8740  C360  203FC1    JSR LINIES ; LINIE ZIEHEN
8750  C363            ;
8760  C363            ; LINIE ( OX , UY , UX , UY )
8770  C363            ;
8780  C363  AD46C0    LDA OX
8790  C366  8D39C0    STA AX
8800  C369  AD47C0    LDA OX+1
8810  C36C  8D3AC0    STA AX+1
8820  C36F  AD4BC0    LDA UY
8830  C372  8D3BC0    STA AY
8840  C375  AD46C0    LDA OX
8850  C378  8D3CC0    STA EX
8860  C37B  AD47C0    LDA OX+1
8870  C37E  8D3DC0    STA EX+1
8880  C381  AD48C0    LDA OY
8890  C384  8D3EC0    STA EY
8900  C387  203FC1    JSR LINIES ; LINIE ZIEHEN
8910  C38A            ;
8920  C38A  60        RTS : ENDE UPRO RECHTECK ZEICHNEN
```

Ebenso wie die Programme PUNKT und LINIE, hat auch dieses Programm zwei Einsprungstellen, die erste mit dem Label RECHT, die vier Parameter aus dem Basic-Text einliest und zwar die Größen OX, OY, UX und UY analog zu ihrer Verwendung in Kapitel 4.2. Die Parameter werden in den namensgleichen Speicherzellen abgelegt, so daß man dorthin die Parameter ablegen muß, wenn man das Label RECHS benützt, das für den Einsatz innerhalb der Maschinenprogramme vorgesehen ist, und somit die zweite Einsprungstelle darstellt.

Nach dem Label RECHS folgen vier analoge Blöcke, die jeweils die Parameter zum Ziehen einer Linie besetzen und dann dieses Unterprogramm aufrufen.

Adresse bei Aufruf mit SYS: 49170

Syntax: RECHT,OX,OY,UX,UY bzw. SYS49170,OX,OY,UX,UY

Kurzes Beispiel:

```
11020 REM * HINTERES RECHTECK *
11025 RECHT,X3,Y3,X2-X1+X3,Y2-Y1+Y3
```

6.2.9 BLOCK

```
9070  C38B             BLOCK:
9080  C38B  20FDAE     JSR CHKCOM
9090  C38E  20EBB7     JSR GETAB   ; PARAMETER SIEHE
9100  C391  8E48C0     STX OY      ; BEI RECHTECK ZEICHNEN
9110  C394  A514       LDA ADRL
9120  C396  8D46C0     STA OX
9130  C399  A515       LDA ADRH
9140  C39B  8D47C0     STA OX+1
9150  C39E  20FDAE     JSR CHKCOM ;KOMMA UEBERLESEN
9160  C3A1  20EBB7     JSR GETAB
9170  C3A4  8E4BC0     STX UY
9180  C3A7  A614       LDX ADRL
9190  C3A9  8E49C0     STX UX
9200  C3AC  A515       LDA ADRH
9210  C3AE  8D4AC0     STA UX+1
9220  C3B1             ;
9230  C3B1             BLOCKS:
9240  C3B1             ; PARAMETER (OX,OY,UX,UY)
9250  C3B1             ; SIEHE RECHTECK
9260  C3B1             ;
9270  C3B1  AD4BC0     LDA UY
9280  C3B4  CD48C0     CMP OY
9290  C3B7  B000   R   BCS BLO1  ; UY > OY
9300  C3B9  AE48C0     LDX OY
9310  C3BC  8E4BC0     STX UY
9320  C3BF  8D48C0     STA OY   ; UY MIT OY VERTAUSCHT
9330  C3C2             BLO1:
9340  C3C2  AE48C0     LDX OY
9350  C3C5  8E45C0     STX YLAUF ; YLAUF = OY
9360  C3C8             BLO2:
9370  C3C8  AD46C0     LDA OX
9380  C3CB  8D39C0     STA AX
9390  C3CE  AD47C0     LDA OX+1
9400  C3D1  8D3AC0     STA AX+1   ; AX = OX
9410  C3D4  AD45C0     LDA YLAUF
9420  C3D7  8D3BC0     STA AY     ; AY = YLAUF
9430  C3DA  8D3EC0     STA EY     ; EY = YLAUF
9440  C3DD  AD49C0     LDA UX
9450  C3E0  8D3CC0     STA EX
9460  C3E3  AD4AC0     LDA UX+1
9470  C3E6  8D3DC0     STA EX+1   ; EX = UX
9480  C3E9  203FC1     JSR LINIES ; LINIE ZIEHEN
9490  C3EC  EE45C0     INC YLAUF  ; YLAUF ERHOEHEN
9500  C3EF  AC4BC0     LDY UY
9510  C3F2  CC45C0     CPY YLAUF  ; ENDE ERREICHT PRINT
9520  C3F5  B0D1       BCS BLO2   ; NEIN
9530  C3F7  60         RTS ; ENDE UPRO BLOCK ZEICHNEN
```

Der Teil zum Einlesen der Parameter des Unterprogramm-Blocks ist praktisch identisch mit dem des Unterprogramms RECHT. Ebenso werden die Eckpunkte in den gleichen Variablenzellen OX, OY, UX und UY angegeben, wenn man das Unterprogramm über das Label BLOCKS aufruft. Zunächst werden die Parameter OY und UY der Größe nach sortiert, so daß später eine Schleife von OY bis UY laufen kann. Diese Schleife verwendet als Laufvariable, die Zelle YLAUF und ruft jeweils das Unterprogramm LINIES auf.

Hier eine Anregung für Fortgeschrittene: Wenn Sie mit diesem Unterprogramm einen Block zeichnen, der einen großen Teil des Bildschirms ausfüllt, so werden Sie merken, daß das Programm noch nicht sonderlich schnell ist. Das liegt vor allem daran, daß jeder Punkt einzeln gesetzt wird. Sie können sich jedoch eine verbesserte LINIE-Routine überlegen, die jeweils acht nebeneinander liegende Punkte gleichzeitig setzt, oder sogar eine spezielle Block-Routine die ganze Zeichenbereiche auf einmal setzt.

Adresse bei Aufruf mit SYS: 49173

Syntax: BLOCK,OX,OY,UX,UY bzw. SYS49173,OX,OY,UX,UY

Kurzes Beispiel:

```
1500 REM *** BLOCK ZEICHNEN ***
1510 BLOCK,150,80,170,100
```

6.2.10 KREIS (bzw. Ellipse)

```
20035  C44D             ELLIPSE:
20040  C44D  20FDAE     JSR CHKCOM
20045  C450  20EBB7     JSR GETAB   ;LIES KX,KY
20050  C453  8E4EC0     STX KY      ;SPEICHERE KY
20055  C456  A514       LDA ADRL
20060  C458  8D4CC0     STA KX
20065  C45B  A515       LDA ADRH
```

```
20070  C45D  8D4DC0    STA KX+1  ;KX GESPEICHERT
20075  C460  20FDAE    JSR CHKCOM ;KOMMA UEBERLESEN
20080  C463  208AAD    JSR FRMNUM :HOLE RX
20085  C466  A0C0      LDY #BASH
20090  C468  A21E      LDX #RX-BASIS
20095  C46A  20D4BB    JSR FACXY
20100  C46D  20FDAE    JSR CHKCOM :KOMMA UEBERLESEN
20105  C470  208AAD    JSR FRMNUM ;HOLE RY
20110  C473  A0C0      LDY #BASH
20115  C475  A223      LDX #RY-BASIS
20120  C477  20D4BB    JSR FACXY
20125  C47A  A91E      LDA #RX-BASIS
20130  C47C  A0C0      LDY #BASH
20135  C47E  2050B8    JSR FKMINUS : FAC = RX-RY
20140  C481  202BBC    JSR FSGN    ; A = SGN(RX-RY)
20145  C484  3000   R  BMI EL1
20150  C486             ; WENN RX >= RY , DANN
20155  C486  A91E      LDA #RX-BASIS
20160  C488  A0C0      LDY #BASH
20165  C48A  20A2BB    JSR FLDFACK : FAC = RX
20170  C48D  4C8080 R  JMP EL2
20175  C490             EL1: ; WENN RX < RY
20180  C490  A923      LDA #RY-BASIS
20185  C492  A0C0      LDY #BASH
20190  C494  20A2BB    JSR FLDFACK : FAC = RY
20195  C497             EL2:
20200  C497  A9BC      LDA #F1L
20205  C499  A0B9      LDY #F1H
20210  C49B  200FBB    JSR FKDIV   : FAC = 1 / FAC
20215  C49E  A22D      LDX #STEP-BASIS
20220  C4A0  A0C0      LDY #BASH
20225  C4A2  20D4BB    JSR FACXY   : STEP = 1/RX BZW. 1/RY
20230  C4A5  A920      LDA #$20
20235  C4A7  8D29C0    STA LAUF+1
20240  C4AA  A900      LDA #$00
20245  C4AC  8D28C0    STA LAUF
20250  C4AF  8D2AC0    STA LAUF+2
20255  C4B2  8D2BC0    STA LAUF+3
20260  C4B5  8D2CCA    STA LAUF+4  : LAUF = 0
20265  C4B8             EL3:
20270  C4B8  A929      LDA #LAUF-BASIS
20275  C4BA  A0C0      LDY #BASH
20280  C4BC  20A2BB    JSR FLDFACK ; FAC = LAUF
20285  C4BF  20EBE2    JSR FSIN    ; FAC = SIN(LAUF)
20290  C4C2  A91E      LDA #RX-BASIS
20295  C4C4  A0C0      LDY #BASH
20300  C4C6  2028BA    JSR FMALK : FAC = SIN(LAUF)*RX
20305  C4C9  A911      LDA #F05L
20310  C4CB  A0BF      LDY #F05H
20315  C4CD  2067B8    JSR FPLUSK : FAC = SIN(LAUF)*RX+0.5
20320  C4D0  209BBC    JSR FLPINT : FAC -> INTEGER
```

```
20325  C4D3  18        CLC
20330  C4D4  A565      LDA INTL
20335  C4D6  6D4CC0    ADC KX
20340  C4D9  8514      STA ADRL
20345  C4DB  A564      LDA INTH
20350  C4DD  6D4DC0    ADC KX+1
20355  C4E0  8515      STA ADRH : ADR = KX+INT(SIN(LAUF)*RX
20360  C4E2  A928      LDA #LAUF-BASIS                +0.5)
20365  C4E4  A0C0      LDY #BASH
20370  C4E6  20A2BB    JSR FLOFACK : FAC = LAUF
20375  C4E9  2064E2    JSR FCOS    : FAC = COS(LAUF)
20380  C4EC  A923      LDA #RY-BASIS
20385  C4EE  A0C0      LDY #BASH
20390  C4F0  2028BA    JSR FMALY   : FAC = COS(LAUF)*RY
20395  C4F3  A911      LDA #F05L
20400  C4F5  A0BF      LDY #F05H
20405  C4F7  2067B8    JSR FPLUSK : FAC = COS(LAUF)*RY+0.5
20410  C4FA  209BBC    JSR FLPINT : FAC -> INTEGER
20415  C4FD  18        CLC
20420  C4FE  A565      LDA INTL
20425  C500  6D4ECO    ADC KY
20430  C503  AA        TAX  ; X = INT(COS(LAUF)*RY+0.5)+KY
20435  C504  2055C0    JSR PUNKS ; PUNKT SETZEN
20440  C507  A928      LDA #LAUF-BASIS
20445  C509  A0C0      LDY #BASH
20450  C50B  20A2BB    JSR FLOFACK ; FAC = LAUF
20455  C50E  A92D      LDA #STEP-BASIS
20460  C510  A0C0      LDY #BASH
20465  C512  2067B8    JSR FPLUSK : FAC = LAUF + STEP
20470  C515  A228      LDX #LAUF-BASIS
20475  C517  A0C0      LDY #BASH
20480  C519  20D4BB    JSR FACXY  : LAUF = LAUF + STEP
20485  C51C  A909      LDA #F2PIL
20490  C51E  A0E3      LDY #F2PIH
20495  C520  2050B8    JSR FKMINUS ; FAC = 2 * PI - LAUF
20500  C523  202BBC    JSR FSGN ; A = SGN( 2*PI - LAUF )
20505  C526  1090      BPL EL3 ; SCHLEIFE BIS LAUF > 2*PI
20510  C528  60        RTS ; ENDE UPRO ELLIPSE ZEICHNEN
```

Diese Routine besitzt vier Parameter von gemischter Art. Der erste Parameter (X-Koordinate des Mittelpunkts) ist ein 2 Byte-Parameter, der nächste Parameter (Y-Koordinate des Mittelpunktes) ist ein Byte-Parameter, und die beiden anderen Parameter (Radius in X- und Y-Richtung) sind Fließkomma-Parameter.

Zum Einlesen eines Fließkomma-Parameters wird die Basic-Routine FRMNUM benützt. Die Parameter werden in die Speicherzellen KX, KX+1, KY sowie 5 Byte in die Fließkomma-Variable RX und 5 Byte in die Fließkomma-Variable RY.

Im weiteren werden im wesentlichen Fließkomma-Routinen verwendet, die den Algorithmus des in Kapitel 4.2.5 beschriebenen Basic-Programms nachvollziehen.

Anschließend müssen noch die erhaltenen Fließkomma-Werte in Integer-Werte umgewandelt werden, um die Koordinaten an das Unterprogramm PUNKT zu übergeben.

Die verwendete Schleife benützt die Fließkomma-Laufvariable LAUF mit der Schrittweite STEP. Die Schleife läuft immer von 0 bis 2 * PI.

Adresse bei Aufruf mit SYS: 49176

Syntax: KREIS,KX,KY,RX,RY bzw. SYS49176,KX,KY,RX,RY

Kurzes Beispiel:

```
1600 REM *** KREIS ZEICHNEN ***
1610 KREIS,200,150,25,10
```

6.2.11 RADIUS

```
21070  C529            RADIUS:
21080  C529  20FDAE    JSR CHKCOM
21090  C52C  20EBB7    JSR GETAB  ;LIES AX,AY
21100  C52F  8E3BC0    STX AY     ;SPEICHERE AY
21110  C532  A514      LDA ADRL
21120  C534  8D39C0    STA AX
21130  C537  A515      LDA ADRH
21140  C539  8D3AC0    STA AX+1   ;AX GESPEICHERT
21150  C53C  20FDAE    JSR CHKCOM ;KOMMA UEBERLESEN
21160  C53F  208AAD    JSR FRMNUM ;HOLE RX
21170  C542  A0C0      LDY #BASH
21180  C544  A21E      LDX #RX-BASIS
21190  C546  20D4BB    JSR FACXY
21200  C549  20FDAE    JSR CHKCOM ;KOMMA UEBERLESEN
21210  C54C  208AAD    JSR FRMNUM ;HOLE RY
21220  C54F  A0C0      LDY #BASH
21230  C551  A223      LDX #RY-BASIS
21240  C553  20D4BB    JSR FACXY
21250  C556  20FDAE    JSR CHKCOM ;KOMMA UEBERLESEN
21260  C559  208AAD    JSR FRMNUM ;HOLE WI
21270  C55C  A0C0      LDY #BASH
21280  C55E  A232      LDX #WINK-BASIS
21290  C560  20D4BB    JSR FACXY
```

```
21300  C563  206BE2   JSR FSIN  ; FAC = SIN(WINK)
21310  C566  A91E     LDA #RX-BASIS
21320  C568  A0C0     LDY #BASH
21330  C56A  2028BA   JSR FMALK ; FAC = SIN(WINK)*RX
21340  C56D  A911     LDA #F05L
21350  C56F  A0BF     LDY #F05H
21360  C571  2067B8   JSR FPLUSK ; FAC = SIN(WINK)*RX+0.5
21370  C574  209BBC   JSR FLPINT ; FAC -> INTEGER
21380  C577  18       CLC
21390  C578  A565     LDA INTL
21400  C57A  6D39C0   ADC AX
21410  C57D  8D3CC0   STA EX
21420  C580  A564     LDA INTH
21430  C582  6D3AC0   ADC AX+1
21440  C585  8D3DC0   STA EX+1 ; EX = AX+INT(SIN(WINK)*RX
21450  C588  A932     LDA #WINK-BASIS                  +0.5)
21460  C58A  A0C0     LDY #BASH
21470  C58C  20A2BB   JSR FLDFACK ; FAC = WINK
21480  C58F  2064E2   JSR FCOS    ; FAC = COS(WINK)
21490  C592  A923     LDA #RY-BASIS
21500  C594  A0C0     LDY #BASH
21510  C596  2028BA   JSR FMALK ; FAC = COS(WINK)*RY
21520  C599  A911     LDA #F05L
21530  C59B  A0BF     LDY #F05H
21540  C59D  2067B8   JSR FPLUSK ; FAC = COS(WINK)*RY+0.5
21550  C5A0  209BBC   JSR FLPINT ; FAC -> INTEGER
21560  C5A3  18       CLC
21570  C5A4  A565     LDA INTL
21580  C5A6  6D3BC0   ADC AY
21590  C5A9  8D3EC0   STA EY :EY=INT(COS(WINK)*RY+0.5)+AY
21600  C5AC  203FC1   JSR LINIES ; LINIE ZIEHEN
21610  C5AF  60       RTS ; ENDE UPRO RADIUS ZEICHNEN
```

Hier werden, ähnlich wie im Unterprogramm KREIS fünf Parameter mit gemischten Typen eingelesen. Das sind: KX, KY, RX und RY, mit der gleichen Bedeutung wie bei KREIS und WI für den Winkel, in dem der Radius gezeichnet werden soll.

In den folgenden Zeilen, in denen wieder einige Aufrufe von Fließkomma-Routinen vorkommen, wird im wesentlichen wieder das Basic-Programm aus Kapitel 4.2.8 nachgebildet.

Adresse bei Aufruf mit SYS: 49179

Syntax: RADIUS,KX,KY,RX,RY,RW bzw. SYS49179,KX,KY,RX,RY,RW

Kurzes Beispiel:

```
100 REM *** 12-ZAEHLIGER STERN ***
110 FOR W= 0 TO 2*π STEP π/6
120 RADIUS,160,100,70,50,W
130 NEXT
```

6.2.12 Symboltabelle

Zum Abschluß der Beschreibung der grafischen Routinen soll noch die Symboltabelle abgedruckt werden, in der alle Variablenzellen, Hilfszellen, Konstanten, ROM-Routinen und Labels zu finden sind.

```
SYMBOLE:
NAME        TYP    WERT
ADRL        CONW   0014
ADRH        CONW   0015
PRDL        CONW   0057
PRDH        CONW   0058
FAK1L       CONW   0059
FAK1H       CONW   005A
FAK2        CONW   005B
QUOTL       CONW   005C
QUOTH       CONW   005D
Z           CONW   00FD
GANFH       CONW   0020
GENDH       CONW   0040
VANFANG     CONW   0400
VIC         CONW   D000
INTL        CONW   0065
INTH        CONW   0064
F1L         CONW   00BC
F1H         CONW   00B9
F05L        CONW   0011
F05H        CONW   00BF
F2PIL       CONW   0009
F2PIH       CONW   00E3
GETBYT      CONW   B79B
GETADR      CONW   B7F7
GETAB       CONW   B7EB
FRMNUM      CONW   AD3A
CHKCOM      CONW   AEFD
ILQERR      CONW   B248
MEMERR      CONW   BB8A
FLOFAC      CONW   BBA2
FPLUSK      CONW   B867
OX          LABL   C046
OY          LABL   C048
UX          LABL   C049
UY          LABL   C04B
KX          LABL   C04C
KY          LABL   C04E
PUNKT       LABL   C04F
PUNKS       LABL   C055
PUN1        LABL   C067
PUN2        LABL   C06F
PUNEND      LABL   C0AC
PUNERR      LABL   C0AD
GREIN       LABL   C0B0
LOESCH      LABL   C0E0
LOE1        LABL   C0E8
LOE2        LABL   C0E9
FARBE       LABL   C0F7
FARB1       LABL   C0FA
GRAUS       LABL   C10C
LINIE       LABL   C119
LINIES      LABL   C13F
LIN1        LABL   C14C
LIN2        LABL   C177
LIN3        LABL   C1AB
LINXGL      LABL   C1BB
LINXGL1     LABL   C1CC
LINXGL2     LABL   C1CF
LINYGL      LABL   C1E7
LINYGL1     LABL   C1FF
LINEND      LABL   C211
LINX        LABL   C212
LINX1       LABL   C21A
LINX2       LABL   C251
```

FSIN	CONW	E26B	LINX3	LABL	C254
FCOS	CONW	E264	LINX4	LABL	C260
FACXY	CONW	BBD4	LINX5	LABL	C272
FCHS	CONW	BFD4	LINY	LABL	C273
FKMINUS	CONW	B850	LINY1	LABL	C278
FMALK	CONW	BA28	LINY2	LABL	C288
FKDIV	CONW	BB0F	LINY3	LABL	C28C
FSGN	CONW	BC2B	RECHT	LABL	C2C8
FLPINT	CONW	BC9B	RECHS	LABL	C2EE
BASH	CONW	0000	BLOCK	LABL	C38B
BASIS	CONW	C000	BLOCKS	LABL	C3B1
RX	LABL	C01E	BLO1	LABL	C3C2
RY	LABL	C023	BLO2	LABL	C3C8
LAUF	LABL	C028	MULT	LABL	C3F8
STEP	LABL	C02D	MULT1	LABL	C401
WINK	LABL	C032	MULT2	LABL	C412
SPEICH1	LABL	C037	MULEND	LABL	C41B
SPEICH2	LABL	C038	DIV	LABL	C41C
AX	LABL	C039	DIV1	LABL	C42D
AY	LABL	C03B	DIV2	LABL	C435
EX	LABL	C03C	DIVEND	LABL	C44C
EY	LABL	C03E	ELLIPSE	LABL	C44D
XDIFF	LABL	C03F	EL1	LABL	C490
YDISGN	LABL	C041	EL2	LABL	C497
YDIABS	LABL	C042	EL3	LABL	C4B8
XLAUF	LABL	C043	RADIUS	LABL	C529
YLAUF	LABL	C045			

6.3 PAUSE - Befehl

Wenn man ein Basic-Programm für kurze Zeit anhalten will, so wird meistens eine Warteschleife (FOR I=1 TO 1000 : NEXT) verwendet. Nachteilig ist, daß eine Variable vergeudet wird, und daß die Zeit nicht in Sekunden angegeben werden kann.

Deshalb wird im folgenden eine Routine vorgestellt, die sich einer der im Commodore 64 eingebauten Uhren bedient, um eine definierte Anzahl von Sekunden zu warten.

```
*** COMMODORE 64   6502-ASSEMBLER ***          VERSION 1.1
ASSEMBLIEREN   VON PAUSE.SRC
OBJECT-DATEI   IST PAUSE.OBJ
SYMBOL-TABELLE IST PAUSE.SYM

ZEILE  ADR.  OBJ     * QUELLTEXT

    10  CA00             .ORG $CA00
```

```
  20  CA00            ;*******************************
  30  CA00            ;*  STARTADRESSE = 51712       *
  40  CA00            ;*******************************
  50  CA00            ;
  60  CA00            ;
  70  CA00            ;
 100  CA00            ;*******************************
 110  CA00            ;* KONSTANTENVEREINBARUNGEN    *
 120  CA00            ;*******************************
 130  CA00            CIA2=$DD00  ; ADRESSE DES CIA2
 131  CA00            ;
 500  CA00            ;*******************************
 510  CA00            ;* BASIC-ROUTINEN              *
 520  CA00            ;*******************************
 525  CA00            ;
 530  CA00            GETBYT=$B79B; BYTE-WERT AUS BASIC-
 531  CA00            ;TEXT IM X-REGISTER UEBERGEBEN
 532  CA00            ;
 540  CA00            STOP=$F6ED; STOPTASTE ABFRAGEN
 541  CA00            ;
1000  CA00            ;*******************************
1010  CA00            ;* PAUSE VON BIS ZU 255 SEC.*
1020  CA00            ;*******************************
1030  CA00            PAUSE:
1040  CA00  AD0EDD    LDA CIA2+14
1050  CA03  0980      ORA #$10000000 ; BIT 7 SETZEN
1060  CA05  8D0EDD    STA CIA2+14 ; AUF 50-HZ-TAKT
1070  CA08            ;
1080  CA08  AD0FDD    LDA CIA2+15
1090  CA0B  297F      AND #$01111111 ; BIT 7 LOESCHEN
1100  CA0D  8D0FDD    STA CIA2+15 ; UHRZEIT FOLGT
1110  CA10            ;
1120  CA10  A203      LDX #3
1130  CA12  A900      LDA #0
1140  CA14            PAU1:
1150  CA14  9D08DD    STA CIA2+8,X ;ZEITREGISTER = 0
1160  CA17  CA        DEX ;NAECHTES REGISTER
1170  CA18  10FA      BPL PAU1
1180  CA1A            ;UHRZEIT IST JETZT 00:00:00:0
1190  CA1A            ;
1200  CA1A  209BB7    JSR GETBYT; PARAMETER HOLEN
1210  CA1D  A900      LDA #0
1220  CA1F  8D8080 R  STA MIN; SOLL-MINUTEN = 0
1230  CA22  F8        SED     ; BCD-ARITHMETIK
1240  CA23            PAU2:
1250  CA23  E000      CPX #0  ;X=0 PRINT
1260  CA25  F000   R  BEQ PAU4;WENN JA,ZUM SCHLEIFENENDE
1270  CA27  CA        DEX ; X HERUNTERZAEHLEN
1280  CA28  18        CLC
1290  CA29  6901      ADC #1 ; 1 IM (BCD)-AKKU ADDIEREN
1295  CA2B  C960      CMP #$60 ; VERGLEICHE MIT 60 (BCD)
```

```
 1300  CA2D  90F4      BCC PAU2 : KEIN UEBERTRAG
 1310  CA2F  EE8080 R  INC MIN ;UEBERTRAG ZU SOLL-MINUTEN
 1315  CA32  A900      LDA #$00 : UND AKKU=0 (SEKUNDEN)
 1320  CA34  F0ED      BEQ PAU2;UNBEDINGTER SPRUNG
 1330  CA36            PAU4:
 1340  CA36  8D8080 R  STA SEC : ALS SOLL-SEK. SPEICHERN
 1350  CA39  D8        CLD :BCD-ARITHMETIK ABSCHALTEN
 1360  CA3A            PAU5:
 1370  CA3A  20EDF6    JSR STOP : STOP-TASTE GEDRUECKT
 1380  CA3D  F000   R  BEQ PAUEND: JA, DANN RTS
 1390  CA3F  AD0ADD    LDA CIA2+10 : MINUTENREGISTER
 1400  CA42  CD8080 R  CMP MIN : MIT SOLLMINUTEN VERGL.
 1410  CA45  90F3      BCC PAU5: NICHT ERREICHT
 1420  CA47  AD09DD    LDA CIA2+9 : SEKUNDENREGISTER
 1430  CA4A  CD8080 R  CMP SEC ; MIT SOLL-SEKUNDEN VERGL.
 1440  CA4D  90EB      BCC PAU5: NICHT ERREICHT
 1450  CA4F            :
 1460  CA4F            PAUEND:
 1470  CA4F  60        RTS
 1480  CA50            :
 1490  CA50            :
 1500  CA50  00        SEC: BYT 0 :HILFSZELLE SEKUNDEN
 1510  CA51  00        MIN: BYT 0 :HILFSZELLE MINUTEN
 1520  CA52            :
59990  CA52            ;*********************************
59991  CA52            ;*                               *
59992  CA52            ;* PROGRAMMENDE                  *
59993  CA52            ;*                               *
59994  CA52            ;*********************************
59999  CA52            .END

 0 FEHLER.
ASSEMBLIERUNG FERTIG.

SYMBOLE:
NAME        TYP   WERT

CIA2        CONW  DD00
GETBYT      CONW  B79B
STOP        CONW  F6ED
PAUSE       LABL  CA00
PAU1        LABL  CA14
PAU2        LABL  CA23
PAU4        LABL  CA36
PAU5        LABL  CA3A
PAUEND      LABL  CA4F
SEC         LABL  CA50
MIN         LABL  CA51
```

Zum Verständnis des Programms ist es notwendig, die Bedeutung der verwendeten Register des CIA (**C**omplex **I**nterface Adapter) im Commodore 64 zu kennen.

Im vorliegenden Programm wird der zweite CIA mit der Startadresse $DD00 = 56576 verwendet. Die Adresse eines Registers ergibt sich also aus Registernummer + $DD00.

CIA - Register 8, 9, 10 und 11

Diese Register enthalten die Zehntel-Sekunden, die Sekunden, die Minuten und die Stunden jeweils im BCD-Format

CIA - Register 14

Bit 7 muß gesetzt werden, wenn der Rechner am 50-Hz-Netz betrieben wird. Ansonsten geht die Uhr nach.

CIA - Register 15

Bit 7 dieses Registers muß gesetzt werden, bevor man die Uhrzeit (Register 8 - 11) setzt.

Im vorliegenden Assembler-Programm wird zunächst die Zeit auf 00:00:00:0 gesetzt, und dann die Anzahl der Sekunden aus dem Basic-Text gelesen (JSR GETBYT). Dann wird mit Hilfe einer Schleife diese Binärzahl in eine BCD-Zahl für die Sekunden und eine für die Minuten umgewandelt. Zur Speicherung dieser Werte werden die Zellen SEC und MIN verwendet.

In der eigentlichen Warteschleife werden nun die CIA-Register 9 und 10 immer wieder abgefragt, bis die Sollzeit abgelaufen ist. Zwischendurch wird aber immer wieder die Stoptaste abgefragt (JSR STOP), damit der Rechner durch eine zu lange Pause nicht völlig blockiert ist.

Adresse für Aufruf mit SYS: 51712

Syntax: PAUSE,SE bzw. SYS 51712,SE

Das Kommando SYS 51712,0 bzw. PAUSE,0 setzt nur die Uhr im CIA 2 zurück. SE bedeutet die Wartezeit in Sekunden, und kann auch eine Variable sein.

6.4 Basic-Funktion ASC() ändern

Es mag vielleicht schon einigen Lesern aufgefallen sein, daß die ASC-Funktion zu einem 'ILLEGAL QUANTITY ERROR' führt, wenn das Argument ein leerer String ("") ist. In vielen Fällen, insbesondere beim Einlesen von einzelnen Zeichen von der Floppy (vgl. Kapitel 5.2) wäre es günstiger, wenn die Funktion ASC("") den Wert '0' liefern würde.

Dazu muß man einen Teil des Basic-Interpreters ändern. Dies funktioniert natürlich nur, wenn der Interpreter im RAM steht. Deshalb müssen wir im Adress-Bereich $A000 bis $BFFF das ROM in das RAM kopieren. In diesem Bereich steht auch die ASC-Funktion.

Der Kopiervorgang kann mit einer Basic-Schleife

```
FOR I = 40960 TO 49151 : POKE I,PEEK(I) : NEXT
```

realisiert werden. Befindet sich der Commodore 64 im Normalzustand, so adressiert man in diesem Bereich (dezimal 40690 bis 49151) mit einem Lesevorgang (PEEK) immer das ROM, und mit einem Schreibvorgang (POKE) immer das RAM. Dadurch gestaltet sich der Kopiervorgang besonders einfach.

Jetzt können wir die Kopie des Basic-Interpreters nach eigenen Vorstellungen abändern. Trotzdem werden immer noch die Basic-Befehle aus dem ROM abgearbeitet, bis wir den Befehl

```
POKE 1,PEEK(1) AND 127
```

geben, womit der Prozessorport Nr. 0 auf Null geschaltet und das ROM ausgeblendet wird.

Betriebssystem-Routinen, die nicht im Bereich $A000 bis $BFFF stehen können mit den obigen Befehlen nicht geändert werden. Wir werden jedoch in einem der folgenden Bände dieser Buchreihe ein Beispiel für eine Betriebssystem-Änderung geben.

Die ASC-Funktion im Basic-Interpreter sieht zunächst so aus:

```
Adr.   Code    Befehl        Kommentar

B78B   2082B7  JSR $B782     String holen, Länge in Y
B78E   F008    BEQ $B798     Wenn Länge=0, dann 'ILLEGAL
                             QUANTITY ERROR '
B790   A000    LDY #$00      Y = 0
B792   B122    LDA ($22),Y   erstes Zeichen holen
B794   A8      TAY           ASCII-Code nach Y
B795   4CA2B3  JMP $B3A2     in Fließkomma umwandeln
B798   4C48B2  JMP $B248     'ILLEGAL QUANTITY ERROR'
```

Wenn der Sprung in Adresse $B78E nach $B795 zeigen würde, wäre die Fehlermeldung abgeschaltet. Es muß jetzt noch sichergestellt sein, daß die Funktion dann wirklich den Wert 0 zurückgibt. Glücklicherweise benötigt die Routine $B3A2 nur das Y-Register als Eingabe-Parameter, und das enthält ja Null, weil darin die Länge des Strings gespeichert war.

Wir brauchen also nur die Zeile bei Adresse $B78E ändern in:

```
B78F   F005    BEQ $B795
```

Die zu ändernde Speicherzelle $B78F hat die dezimale Adresse 46991, so daß die gesamte Änderung mit dem Befehl

```
POKE 46991,5
```

erledigt werden kann. Hier noch einmal das Listing des gesamten Basic-Programms, das für die Änderung des ASC-Befehls benötigt wird:

```
50 FORI=40960TO49151
60 POKEI,PEEK(I)
70 NEXT                    :REM BASIC IN RAM KOPIEREN
80 POKE46991,5             :ASC-FUNKTION AENDERN
90 POKE1,PEEK(1)AND127     :KOPIE EINSCHALTEN
```

6.5 Ein- und Ausschalten der Basic-Erweiterungen

Beim Einschalten der Basic-Befehle gibt es folgende Punkte zu beachten:

Die Maschinenprogramme für die Basic-Erweiterungen stehen zunächst noch auf Floppy und müssen deshalb erst geladen werden. Dann muß noch das Maschinenprogramm INIT aufgerufen werden, das den Befehls-Dekodier-Vektor umsetzt.

Im Direktmodus gibt man dazu folgende Befehle ein:

```
LOAD "BASERW.OBJ",8,1
NEW
LOAD "GRASS.OBJ",8,1
NEW
LOAD "PAUSE.OBJ",8,1
NEW
SYS 51200
```

Damit sind alle oben beschriebenen Befehle (mit Ausnahme der geänderten ASC-Funktion) eingeschaltet.

Will man die Erweiterungen in einem Basic-Programm einschalten, so muß dies etwa folgendermaßen geschehen:

```
10 IF L=0 THEN L=1 : LOAD "BASERW.OBJ",8,1
20 IF L=1 THEN L=2 : LOAD "GRASS.OBJ",8,1
30 IF L=2 THEN L=3 : LOAD "PAUSE.OBJ",8,1
40 SYS 51200
```

Da das Programm nach einem LOAD-Befehl immer wieder zur ersten Zeile springt, muß in einer Variablen (hier: L) festgehalten werden, welche Maschinenprogramme schon geladen worden sind.

Diese Verfahrensweise mag etwas umständlich erscheinen, jedoch können hier zusätzliche Erweiterungen leichter eingebaut werden, als wenn alle Befehle in einer einzigen Objekt-Datei gespeichert sind.

Das Ausschalten geschieht einfach durch den neuen Befehl

```
AUS          bzw. SYS 51211
```

der den Befehls-Dekodier-Vektor wieder auf seinen ursprünglichen Wert zurücksetzt.

Bei rechenzeitintensiven Abläufen innerhalb komplexer Programme in denen keine Basic-Erweiterungen benötigt werden, kann man die Erweiterungen beliebig mit SYS 51200 und AUS ein- bzw. abschalten, um den Umweg der Prüfung von neuen Befehlen einzusparen. Dies ist jedoch nur in Extremfällen notwendig.

SYS 51200 kann man beliebig oft eingeben, ohne weitere Auswirkungen zu erzielen. Bei komplexen Programmstrukturen ist jedoch besonderes Augenmerk auf den Befehl AUS zu legen, der offensichtlich nur einmal nach SYS 51200 gegeben werden kann.

7

Sonstige Tips und Tricks

7. Sonstiges

Im letzten Kapitel sollen noch einige kleinere Hinweise zur besseren Nutzung des Commodore 64 gegeben werden. Da Programme der alten Serien 2000/3000 übernommen werden können, und als reines Basic-Programme im Prinzip sofort lauffähig sind, soll kurz auf die Übernahme solcher Programme eingegangen werden.

Für die Serie 8000 gibt es mehr programmtechnische Hilfsmittel, so daß eine Programmeditierung dort auch für Commodore 64-Programme sinnvoll ist.

Ganz zum Schluß noch einige Hinweise auf besondere Eigenheiten und die wichtigsten Adressen aus der Zero-Page.

7.1 Programmübernahme über Kassettenrekorder

Die Übernahme von Basic-Programmen der Serien 3000/4000/8000 auf den Commodore 64 ist unproblematisch, da der Commodore 64 die Programme selbst anpasst.

7.1.1 Serie 2000 auf Commodore 64

Die Programmdateien beim Commodore 64 sind gegenüber der 2000er Serie um ein Byte kürzer. Deshalb ist mit den Anweisungen (im Direkt-Modus)

```
POKE 2051,PEEK(2052)
POKE 2052,PEEK(2053)
POKE 2053,58
```

schon die Lauffähigkeit hergestellt. Dies gilt natürlich nur für reine Basic-Programme. In den ersten beiden Zeilen wird die Anfangsadresse des Basic-Programmes um eine Adresse vorgezogen, und in der dritten Zeile wird das 'frei' gewordene Zeichen in der ersten Programmzeile mit einem ':' besetzt. Diese Doppelpunkt kann man dann im Basic-Editor nach LIST ganz normal löschen, wenn er stört.

7.1.2 Commodore 64 auf Serien 8000 und 3000

Bei Übernahme von Programmen auf Rechner der Serie 8000 und 3000 sieht die Sache etwas anders aus. Nach dem Laden des Programmes über den Kassettenrekorder ruft man zunächst den eingebauten Monitor mit

```
SYS 1024
```

auf, und läßt sich die Speicherzellen 1024 bis 1030 (dezimal) anzeigen mit

```
.m 0400 0400
```

Anschließend ändert man die Anzeige wie folgt ab:

```
. 0400 00 01 08 00 00 3A 00 aa
```

Damit wird der Programmanfang auch für die 8000/3000er Serie auf das beim Commodore 64 übliche 2. Kilobyte des RAMs gelegt, und eine Zeile 0 eingefügt, die einen ':' enthält.

Dann kann man mit

```
.x
```

den Monitor verlassen und mit

```
0       (Return-Taste)
```

die Zeile 0 gleich wieder löschen. Mit

```
LIST
```

hat man nun das gleiche Programm wie auf dem Commodore 64.

7.2 Interessante Kleinigkeiten

Rund um den Cursor

Aktuelle Cursorposition

Die aktuelle Position des Cursors läßt sich mit

```
PEEK(214)       (für die Zeile)
```

```
PEEK(211)       (für die Spalte)
```

abfragen. Ebenso können Sie mit

```
POKE 214,(Zeile)
POKE 211,(Spalte)
```

den Cursor setzen. Damit haben wir zwar den Cursor (für den Rechner) gesetzt, jedoch wissen Bildschirm- und Farb-RAM davon noch nichts. Aber dafür gibt es im Betriebssystem eine Routine, die man mit

```
SYS 58460
```

aufrufen kann. Als Übung für den Leser mögen die Routinen aus Kapitel 2.1 und 2.2 dienen, womit die Unterprogramme ab Zeile 20000 entfallen können.

ACHTUNG !! Spalten und Zeilen werden von 0-39 und 0-24 numeriert !

Cursor ein- bzw. ausschalten

Der Cursor blinkt in der Regel immer, wenn der Computer etwas vom Anwender wissen will, außer wenn ein Zeichen von der Tastatur mittels des GET-Befehls angefordert wird. Soll der Cursor auch bei einer Eingabe mit GET blinken, so ist das Flag in der Zero-Page (Zellen 0-255 im RAM) entsprechend zu ändern. '0' bedeutet 'Cursor ein' und ein anderer Wert 'Cursor aus'. Also:

```
POKE 204,0      : REM Cursor ein
POKE 204,1      : REM Cursor aus
```

Ob der Cursor gerade blinkt oder nicht, kann man durch

```
PEEK(207)
```

erfragen. Hier bedeutet der Wert '0' allerdings Cursor aus. Um Fehler zu vermeiden, sollte man vor dem Einschalten den Cursor erst auf 'dunkel' stellen, also die Befehlsfolge

```
POKE 207,0 : POKE 204,0
```

eingeben.

REPEAT aller Tasten (Dauerfunktion)

In der Regel sind nur drei Tasten (fünf Funktionen) mit einem automatischen Repeat ausgerüstet: die Cursorfunktionen und die Leertaste. Für alle anderen Tasten ist das erste Bit in Zellen 650 des RAM interessant. Ist dieses Bit gesetzt (PEEK(650) größer gleich 128), so haben alle Tasten eine Dauerfunktion:

```
POKE 650,128        : REM Dauerfunktion ein
POKE 650,0          : REM Dauerfunktion aus
```

WAIT-Befehl

Anstatt einer Schleife

```
100 GET A$: IF a$ = "" THEN 100
```

kann man auch schreiben

```
WAIT 203,63 oder
WAIT 198,1
```

Der erste Befehl wartet, bis eine Taste gedrückt wird und der zweite Befehl wartet, bis die Zahl der Zeichen im Tastaturpuufer 1 ist. Beide Befehl haben die gleiche Wirkung.

An den beiden Beispielen merkt man schon die Wirkungsweise des Befehls: es wird gewartet, bis in der zuerst genannten Speicherzelle der Wert erscheint, der hinter dem Komma eingegeben wurde. Dies gestattet eine elegantere Programmierung, als die anfangs vorgestellte Schleife. Aber sie hat auch Nachteil, denn die Kompatibilität zu Rechnern einer anderen Serie geht verloren, da das RAM des Betriebssystems bei jedem Rechner anders aufgebaut ist.

Weitere WAIT-Befehle:

```
WAIT 653,1          :REM Warten auf die SHIFT-Taste
WAIT 653,2          :REM Warten auf die C= -Taste
WAIT 653,4          :REM Warten auf die CTRL-Taste
```

7.3 Die wichtigsten Adressen aus der Zero-Page

Hex-Adresse	Dezimal	Bedeutung
2B - 2C	43 - 44	Zeiger auf Basic-Programm-Start (L,H)
2D - 2E	45 - 46	Zeiger auf Beginn der Variabeln (L,H)
2F - 30	47 - 48	Zeiger auf Beginn der Arrays (L,H)
31 - 32	49 - 50	Zeiger auf Ende der Arrays (L,H)
33 - 34	51 - 52	Zeiger auf Beginn der Zeichenreihen (L,H)
37 - 38	55 - 56	Zeiger auf Ende des Basic-RAM (L,H)
73 - 8A	115 - 138	CHRGET-Routine für Zeichen aus Basic-Text
7A - 7B	122 - 123	Programmzeiger (L,H)
90	144	ST (Status)
91	145	Flag für Stop-Taste
98	152	Anzahl der offenen Dateien
C5	197	Letzte gedrückte Taste
C6	198	Anzahl der Zeichen im Tastaturpuffer
C7	199	Flag für RVS-Modus
CB	203	Gedrückte Taste
D3	211	Aktuelle Spalte des Cursors
D6	214	Aktuelle Zeile des Cursors
---	---	---
0100 - 01FF	256 - 511	Stack des Rechners
0277 - 0280	631 - 640	Tastaturpuffer
0281 - 0282	641 - 642	Start Basic-RAM
0283 - 0284	643 - 644	Ende Basic-RAM
0286	646	Aktuelle Farbe
0287	647	Farbe unter Cursor
0289	649	Länge Tastaturpuffer
028A	650	REPEAT alle Tasten
028D	653	Flag für SHIFT, C= und CTRL Bit 0, 1 und 2
028E	654	Flag für SHIFT
02C0 - 02FE	704 - 766	Block 11 (für Sprites)
0300 - 0301	768 - 769	Vektor für Fehlermeldung
0311 - 0312	785 - 786	USR-Vektor (normal:$B248)
0314 - 0315	788 - 789	IRQ-Vektor (normal:$EA31)

Hex-Adresse	Dezimal	Bedeutung
0316 - 0317	790 - 791	BRK-Vektor (normal:$FE66)
0318 - 0319	792 - 793	NMI-Vektor (normal:$FE47)
033C - 03FB	828 -1019	Bandpuffer
0340 - 037E	832 - 894	Block 13 (für Sprites)
0380 - 03BE	895 - 958	Block 14 (für Sprites)
03C0 - 03FE	960 -1022	Block 15 (für Sprites)

Dort wo (L,H) steht, werden Werte bis 65535 gespeichert, die wie folgt zu errechnen sind:

Z = PEEK (1.Zelle) + 256 * PEEK (2.Zelle)

Anhang

Anhang 1

Tabelle der Parametertypen:

A - Ausgabeparameter von diesem Unterprogramm
E - Eingabeparameter für dieses Unterprogramm
G - Globale Variable
H - Hilfsvariable
P - Aufrufparameter an Unterprogramm
R - Rückgabeparameter von Unterprogramm
T - Transienter Parameter (ist in einem Unterprogramm gleichzeitig Eingabeparameter (E) und Aufrufparameter (P) bzw. A und R

Globale Variablen gibt es zwar bei Basic nicht, da Basic keine block-orientierte Sprache wie z.B. PL/1 ist, aber in diesem Buch ist dieser Begriff für Variablen verwendet worden, die man in anderen Sprachen global definiert hätte.

Da der verwendete Typenraddrucker die Zeichen 'größer als' und 'kleiner als' nicht drucken kann, wurden diese Zeichen wie folgt ersetzt:

NE - Not Equal	/ ungleich
GT - Greater Than	/ größer als
LT - Less Than	/ kleiner als
GE - Greater or Equal	/ größer gleich
LE - Less or Equal	/ kleiner gleich

Die in den Kapiteln angegebenen Zeilenbereiche können mit denen der abgedruckten Listings differieren, weil offensichtliche REM-Zeilen (in der Regel die Überschriften von Unterprogrammen) nicht mitgedruckt wurden.

Bei den Programmen mit hochauflösender Grafik wurden - entgegen dem abgedruckten Listing - einige REM-Zeilen gelöscht, da sonst der für hochauflösende Grafik reservierte Basic-Programm-Bereich überschritten worden wäre!

Anhang 2

Die Farbcodes des Commodore 64:

FARBE	POKE-WERT	TASTE	ZEICHEN
SCHWARZ	0	CTRL 1	
WEISS	1	CTRL 2	
ROT	2	CTRL 3	
TUERKIS	3	CTRL 4	
VIOLETT	4	CTRL 5	
GRUEN	5	CTRL 6	
BLAU	6	CTRL 7	
GELB	7	CTRL 8	
ORANGE	8	C= 1	
BRAUN	9	C= 2	
HELLROT	10	C= 3	
GRAU 1	11	C= 4	
GRAU 2	12	C= 5	
HELLGRUEN	13	C= 6	
HELLBLAU	14	C= 7	
GRAU 3	15	C= 8	

Anhang 3

In den Programmlistings tauchen folgende Bildschirmsteuerzeichen auf:

BILDSCHIRMSTEUERCODES

ZEICHEN	CODE
CURSOR NACH UNTEN	17
CURSOR NACH OBEN	145
CURSOR NACH RECHTS	29
CURSOR NACH LINKS	157
DEL (ZEI. LOESCHEN)	20
HOME (CRSR OBEN LINKS)	19
CLR (BILDSCHIRM LOESCHEN)	147
REVERSE EIN	18
REVERSE AUS	146

Anhang 4

Sprite-Register: (Basis 53248 ist jeweils zu addieren)

Ganze Register je Sprite:

Sprite	X	Y	Farbe
0	0	1	39
1	2	3	40
2	4	5	41
3	6	7	42
4	8	9	43
5	10	11	44
6	12	13	45
7	14	15	46

Gesamtregister mit einem Bit je Sprite:

Überträge X-Koordinaten	16
Sprite ein/aus	21
Ausdehnung in X-Richtung	23
Ausdehnung in Y-Richtung	29
Priorität gegen Hintergrund	27
Multicolor	28
Kollision Sprite/Sprite	30
Kollision Sprite/Hintergrund	31

Ausblick auf die weiteren Bände:

Band 2: Basic-Spiele

Spiele * Spiele * Spiele * Spiele * Spiele * Spiele *

Denkspiele, Kartenspiele, Börsenspiele, Wirtschaftsspiele, ...

Spiele * Spiele * Spiele * Spiele * Spiele * Spiele *

Ein Buch nicht nur zum Spiele abtippen, denn die gibt es auch auf Diskette, sondern zum selbst ergänzen und ändern. Alle Spiele sind in der gelieferten Form für komplettes Spiel geeignet, aber Software läßt sich ja immer verbessern.

Band 3: Leitfaden für Fortgeschrittene

- Multi-Color-Sprites
- Multi-Color-Grafik
- Sound-Generator
- Disassembler
- Datenverwaltung mit der Floppy
- Deutsche Fehlermeldungen

Band 4: Assembler / Disassembler

Der Assembler aus Band 1 und der Disassembler aus Band 3 werden um einiges ergänzt, z.B. durch Aneinanderbinden von Source-Files, und alles wird natürlich genauer erklärt.

Band 5: Simon's-Basic

Nützliches im Umgang mit Simon's Basic. Ein erweitertes und kommentiertes Handbuch.

Band 6: Spiele

Nochmals Spiele. Diesmal mit Maschinenprogrammen und Musik sowie vielen Kniffen.

Lieferbare Markt & Technik-Titel:

Programme und Tips für VC-20
Best.-Nr. MT 513 DM 38,—*

Das VC-20 Buch
Best.-Nr. MT 516 DM 49,—*

Das VC-20 Buch — Beispiele auf Kassette
Best.-Nr. MT 581 DM 19,90*

Das VC-20 Buch — Beispiele auf Diskette
Best.-Nr. MT 582 DM 29,90*

Ein-Chip-Mikrocomputer-Handbuch
Best.-Nr. MT 517 DM 58,—*

Das Datenbanksystem dBASE II
Best.-Nr. MT 524 DM 68,—*

Basic-80 und CP/M
Best.-Nr. MT 525 DM 48,—*

Einführung in Datenbanksysteme mit dBASE II
Best.-Nr. MT 526 DM 68,—*

dBASE II richtig eingesetzt
Best.-Nr. MT 541 DM 68,—*

dBASE II richtig eingesetzt — Beispiele auf Diskette (IBM-PC)
Best.-Nr. MT 544 DM 30,—*
ab 1.1.84 DM 48,—*

Einführung in C
Best.-Nr. MT 561 DM 69,—*

Mit Lotus 1-2-3 zur integrierten Problemlösung
Best.-Nr. MT 562 DM 68,—*

Basic-Dialekte im Vergleich
Best.-Nr. MT 564 DM 32,—*

Das Commodore 64-Buch, Bd. 1: Leitfaden für Erstanwender — mit Assembler
Best.-Nr. MT 591 DM 48,—*

Das Commodore 64-Buch, Bd. 1: Beispiele auf Diskette
Best.-Nr. MT 592 DM 58,—*

Das Commodore 64-Buch, Bd. 2: Basic-Spiele
Best.-Nr. MT 593 DM 38,—*

Das Commodore 64-Buch, Bd. 2: Beispiele auf Diskette
Best.-Nr. MT 594 DM 58,—*

Das Commodore 64-Buch, Bd. 3: Leitfaden für Fortgeschrittene
Best.-Nr. MT 595 DM 38,—*

Das Commodore 64-Buch, Bd. 3: Beispiele auf Diskette
Best.-Nr. MT 596 DM 58,—*

Das Commodore 64-Buch, Bd. 4: Leitfaden für Programmierer — Assembler — Disassembler
Best.-Nr. MT 597 DM 38,—*

Das Commodore 64-Buch, Bd. 4: Beispiele auf Diskette
Best.-Nr. MT 598 DM 58,—*

Das Commodore 64-Buch, Bd. 5: Simon's Basic
Best.-Nr. MT 599 DM 38,—*

Das Commodore 64-Buch, Bd. 5: Beispiele auf Diskette
Best.-Nr. MT 600 DM 58,—*

Computerspiele und Wissenswertes — Commodore 64
Best.-Nr. MT 601 DM 29,80*

Computerspiele und Wissenswertes — Commodore 64: Beispiele auf Diskette
Best.-Nr. MT 602 DM 38,—*

Das Spielebuch Commodore 64
Best.-Nr. MT 603 DM 29,80*

Das Spielebuch Commodore 64: Beispiele auf Diskette
Best.-Nr. MT 604 DM 38,—*

Software-Schnellkurs: CP/M
Best.-Nr. MT 605 DM 37,—*

Software-Schnellkurs: MailMerge
Best.-Nr. MT 606 DM 37,—*

Software-Schnellkurs: dBASE II
Best.-Nr. MT 607 DM 37,—*

Software-Schnellkurs: SuperCalc
Best.-Nr. MT 608 DM 37,—*

Software-Schnellkurs: WordStar
Best.-Nr. MT 609 DM 37,—*

Software-Schnellkurs: Multiplan
Best.-Nr. MT 610 DM 37,—*

Software-Schnellkurs: Lotus 1-2-3
Best.-Nr. MT 611 DM 37,—*

* inkl. MwSt. zuzügl. Versandkosten

Markt&Technik

Hans-Pinsel-Straße 2 · 8013 Haar bei München · Telefon 4613-220

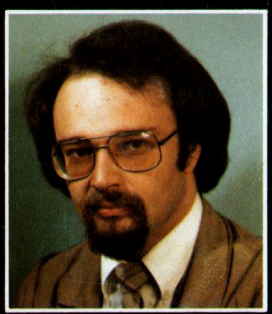
Das Commodore 64-Buch

Band 1: Ein Leitfaden für den Erstanwender

Einige Bücher, die den Commodore 64 zum Thema haben, sind schon auf dem Markt erhältlich. Die meisten Bücher sind jedoch für die Belange von bestimmten Anwendergruppen geschrieben.
Das vorliegende Buch soll für den Einsteiger mit dem Commodore 64 ein Leitfaden sein. Das heißt nicht, daß dieses Buch einen kompletten Basic-Kurs beinhaltet, sondern daß von Beginn an alle Möglichkeiten des Commodore 64 beschrieben werden. Im ersten Kapitel werden zur Einführung einige Programme beschrieben, die in dieser Form auch auf anderen Rechnern (besonders anderen Commodore-Rechnern) laufen. Erst nach dieser Grundlage wird auf die speziellen Eigenschaften des Commodore 64 eingegangen.
Die meisten Programme sind so aufgebaut, daß sie vom Leser noch nach seinen Wünschen ergänzt werden können. Für Übungszwecke werden immer wieder Tips für Änderungen oder Erweiterungen gegeben. Es ist nicht Sinn dieses Buches, vorgefertigte Programme zu liefern, sondern den Leser beim aktiven Arbeiten mit seinem Rechner zu unterstützen.

Die vielfältigen Möglichkeiten des Commodore 64 konnten nicht alle in diesem Buch behandelt werden, da großer Wert auf eine genaue, leichtverständliche Beschreibung der Fakten gelegt wurde. Es sei daher an dieser Stelle auf die weiteren Bände hingewiesen.

Band 2 dieser Reihe befaßt sich mit Spielen und ist analog zu Band 1 aufgebaut. Also nicht nur »Diskette rein — spielen«, sondern »spielend Programmieren lernen«. Band 3 ist ein Leitfaden für Fortgeschrittene und behandelt die Themen, die in Band 1 nicht behandelt wurden. Band 4 ist den Freunden des Maschinenprogramms gewidmet. Dort werden ein ausführlicher Assembler und ein Disassembler genau erklärt. Band 5 ist einer weitverbreiteten Software-Erweiterung des Commodore 64 gewidmet: dem Simon's Basic. Band 6 bringt schließlich wieder Spiele — für Spiele-Profis mit Maschinenprogrammen. Zusätzlich sind zu jedem Band Disketten erhältlich, die dem Leser das Eintippen der beschriebenen Beispiele ersparen.



HANS LORENZ SCHNEIDER
geboren am 15.10.1953 in Köln. Nach dem Abitur 1973 studierte er von 1976 bis 1980 Informatik an der Bundeswehrhochschule in München. Seit 1980 ist Schneider Inhaber und Geschäftsführer eines Software-Hauses, das sich hauptsächlich mit der Erstellung von Individual-Software für Mikrocomputer (Commodore) befaßt.

WERNER EBERL
geboren am 23.2.1962 in München, begann gleich nach dem Abitur 1980 sein Physik-Studium. Seine große Leidenschaft waren schon immer die Computer. Seit 1980 ist er auch als freier Mitarbeiter für ein Software-Haus tätig, wo er für die Umsetzung von Konzepten in lauffähige Programme verantwortlich ist.

Markt&Technik Verlagsgesellschaft mbH
ISBN 3-922120-61-X